Operator's Handbook

ObjectBench System Control Software

Issue 14

August 2000

File reference: OBPROG14

Oxford Instruments
Superconductivity
Tubney Woods, Abingdon,
Oxon, OX13 5QX, England
Tel: +44 (0)1865 393 200
Fax: +44 (0)1865 393 333
E-mail:superconductivity@oxinst.co.uk
www.oxford-instruments.com

Contents

1 Introduction

Oxford Instruments ObjectBench is a general purpose program for controlling instrumentation from a PC compatible computer. It makes use of the features of Microsoft Windows^{TM} to provide a very flexible yet simple user interface. ObjectBench uses standard hardware interfaces, such as RS232, Oxford Instruments ISOBUS or GPIB, to control instrumentation remotely. Sequences, called macros, may be set up to perform a series of operations and data acquisitions. Data acquired in this way may be displayed graphically, printed, and stored in data files.

This manual describes how to install ObjectBench and how to use it for controlling specific instruments. It does not describe how to install Windows or DOS or how to set up your computer - these are fully described in the PC's manuals. Neither does it explain manual operation of instruments; this is described in specific manuals for those instruments.

This manual does not contain safety information for the instruments or systems that this software can be used to control. For this information, please refer to individual instrument manuals and system documentation.

1.1 Features

Here is a brief summary of ObjectBench's features:

- Remote control of electronic instruments supplied by Oxford Instruments using a software front panel which closely mimics the real front panel.
- The ability to monitor signals derived from the instruments in a very flexible way. Data may be displayed graphically and stored in data files.
- Sequences of operations and measurements may be set up using a simple BASIC-like macro language. Generic serial and GPIB commands are provided for third party instruments.
- Data may be stored in ASCII text files, and retrieved for analysis or printing.
- Complete named system configurations may be saved so that a number of different experiments may be stored and retrieved separately.
- Complete customised dialog boxes may be created by the user for calling named macros. This requires the use of Borland's Resource Workshop which is available from Borland International Inc.

It should not be possible to damage Oxford Instruments electronic instruments using this software. However, it is possible that these instruments could be mis-used in such a way that the system is damaged.

1.2 Hardware Requirements

ObjectBench requires a PC compatible computer which is capable of running Microsoft Windows. It is recommended that the PC has the following minimum specifications:

- 486 or higher processor
- colour VGA or super VGA graphics
- 4 megabytes of RAM
- 60 megabyte hard disk
- mouse
- high density 3.5" floppy disk drive

If you are using a 386 or 486sx we recommend that you use a co-processor.

The following are optional but of benefit:

- a printer or plotter supported by Windows

The PC should have installed on it DOS and Windows 3.1, 3.11, Windows 95 or Windows 98 before ObjectBench is installed.

2 Installing ObjectBench

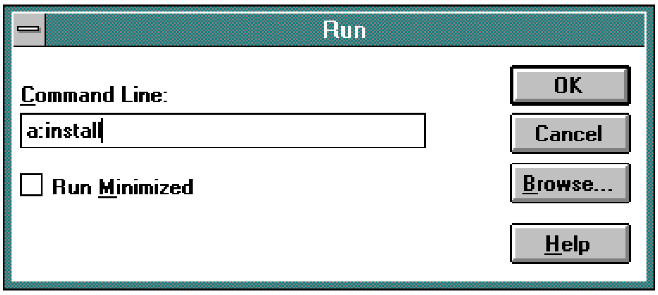
Before attempting to install ObjectBench, make sure your PC has installed on it already DOS and Windows.

If Windows is not already running, run it thus:

win + **<ENTER>**

Maximise the Program Manager by double clicking on it (refer to the Windows manual for more information on the Program Manager). Insert the installation disk in the floppy disk drive and use the Program Manager to start installation as follows.

Type **alt-f, r** to access the "Run..." menu option. A dialog box will be displayed which invites you to type the name of the installation program. In this case, type "a:install.exe" where a: is the floppy disk drive identifier. See Figure 1.



Run

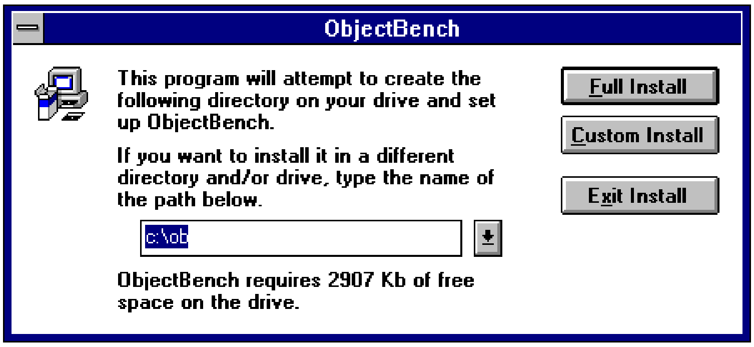
Command Line:

a:install

□ Run Minimized

Figure 1 Running the ObjectBench Installation program.

Click OK. At this point the installation program will run. A dialog similar to that shown in Figure 2 will appear.



ObjectBench

This program will attempt to create the following directory on your drive and set up ObjectBench.

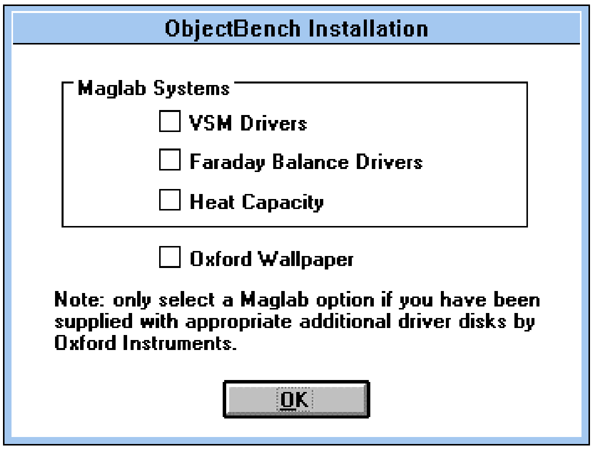
If you want to install it in a different directory and/or drive, type the name of the path below.

c:\ob

ObjectBench requires 2907 Kb of free space on the drive.

Figure 2 The first installation dialog.

Name the directory to install ObjectBench in, or accept the default value of "c:\ob". Then click on "Select Options" to display the dialog shown in Figure 3. If you are installing ObjectBench as part of a Faraday Balance, Vibrating Sample Magnetometer (VSM) system or Heat Capacity system, select the appropriate option by clicking next to it. Later on you will need an extra installation disk if you select either of these options. If you want the Oxford Instruments wallpaper to appear as the background to your Windows desktop, select that option too. Finally click on OK.



ObjectBench Installation

Maglab Systems

□ VSM Drivers

□ Faraday Balance Drivers

□ Heat Capacity

□ Oxford Wallpaper

Note: only select a Maglab option if you have been supplied with appropriate additional driver disks by Oxford Instruments.

Figure 3 - The Installation Options Dialog

The installation program will prompt you to insert any further floppy disks that may be required, and finally it will ask you if you wish to add an ObjectBench program group and icons to the file manager. If this is the first installation of ObjectBench, reply "Yes". If you are upgrading a previous version, reply "No".

When the installation has been successfully completed, you will find that a new program group and icon have been added to the Windows Program Manager.

3 Running ObjectBench

To run ObjectBench, open the Windows Program Manager, open the ObjectBench program group, and double click on the ObjectBench Icon, as shown in Figure 4.

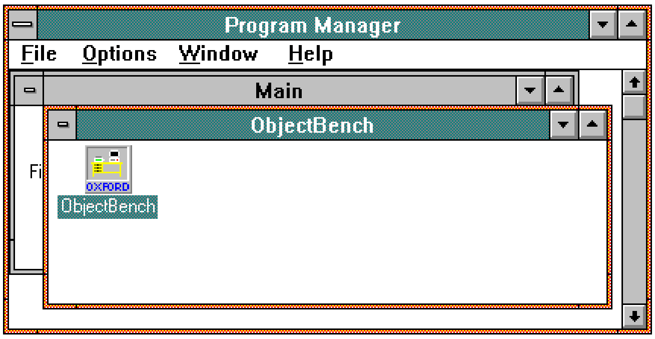


Figure 4 Running ObjectBench from the Windows Program Manager.

ObjectBench will now run, and display its main window, thus:

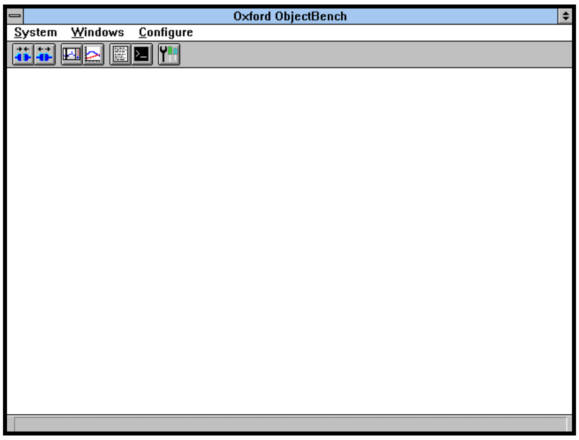


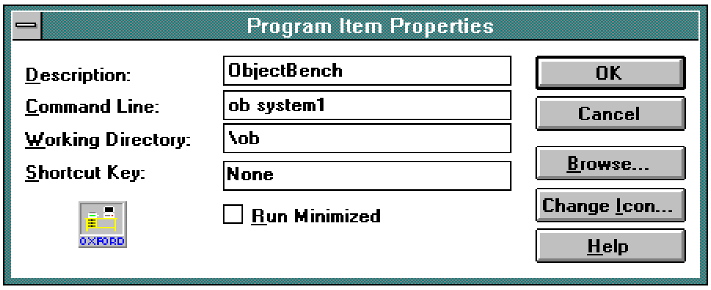
Figure 5 ObjectBench Main Window.

3.1 Running ObjectBench with a Named Setup

When ObjectBench is run, it first searches for a setup file with the file name extension ".obe". This file contains information about the screen layout and values of many setup parameters, so that ObjectBench "remembers " the settings used last. This file is saved when ObjectBench is exited.

If there is more than one common setup in use, more than one setup file can be used so that each setup is stored separately. The name of the setup file must be supplied on the command line used to run ObjectBench.

For example, suppose ObjectBench is frequently used to run two different experiments. A second ObjectBench icon could be added to the Program Manager (use the File/New menu option on Program Manager), and the command line options added by clicking **once** on an ObjectBench icon and typing **alt-<ENTER>.** A dialog box similar to Figure 6 will be displayed.



Program Item Properties

Description: ObjectBench

Command Line: ob system1

Working Directory: \ob

Shortcut Key: None

□ Run Minimized

Figure 6 Editing the ObjectBench icon's properties in the Program Manager.

Add to the end of the command line field the name of the setup to use, in this example "system1". Similarly, "system2" would be added to the end of the command line for the other ObjectBench icon. Change the description to identify the two icons uniquely. These two ObjectBench programs will now run using independent setup files. ObjectBench automatically adds ".obe" to the end of the setup name to create a file name.

4 Using ObjectBench

After running ObjectBench for the first time, the main window be displayed similar to that shown in Figure 5. The main window consists of a menu bar, a button bar and a button window. All ObjectBench functions can be accessed using the menus. Some frequently used functions can also be accessed by clicking on one the buttons below the menu, or on the separate button window. The separate button window is always displayed on top of any other windows on the desktop. The button bar and button window can be removed if preferred, by using the “Configure”/”View button bar” and “Configure”/”View button window” menu options. The function of each button is described in section 4.1.

To exit from ObjectBench, double click on the system menu symbol (the grey square to the left of the main ObjectBench title). On exiting, the setup options are saved so that next time ObjectBench is run, the same windows with the same settings are present.

4.1 Button Functions

This section describes the function of each button on the button bar.

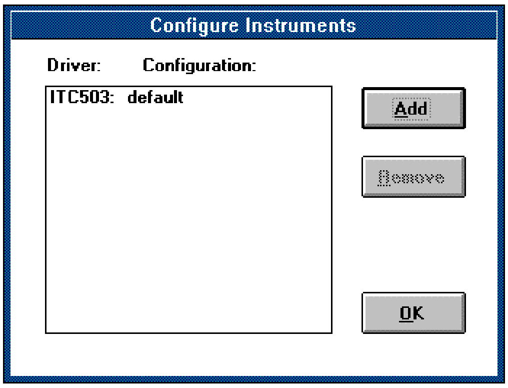
	Equivalent Menu Item	Function
	“System”/”Connect All”	Connect all instruments and XY plots.
	“System”/”Disconnect All”	Disconnect all instruments and XY plots.
	“Windows”/”XY Plot”	Open a new XY plot window.
	“Windows”/”Graphical Analysis”	Open a new analysis window.
	“Windows”/”Edit macro”	Open a new editor window.
>_	“Windows”/”Macro shell”	Open a new macro shell.
	“Windows”/”Macro engineer”	*Open a Macro Engineer window.

* Note that the Macro Engineer is available only in versions of ObjectBench supplied with an Optistat system. In all other cases it is greyed out.

4.2 How to Control an Instrument

4.2.1 Loading an Instrument Driver

To control an instrument it is necessary to load a corresponding instrument driver. This is accomplished by selecting the "Add/Remove Instruments" option from the "Configure" menu. At this point, a dialog box will be displayed listing any instrument drivers that are already loaded. Figure 7 shows a typical Configure Instruments dialog, in which there is a driver for the ITC503 temperature controller already loaded, with a configuration name of "default".

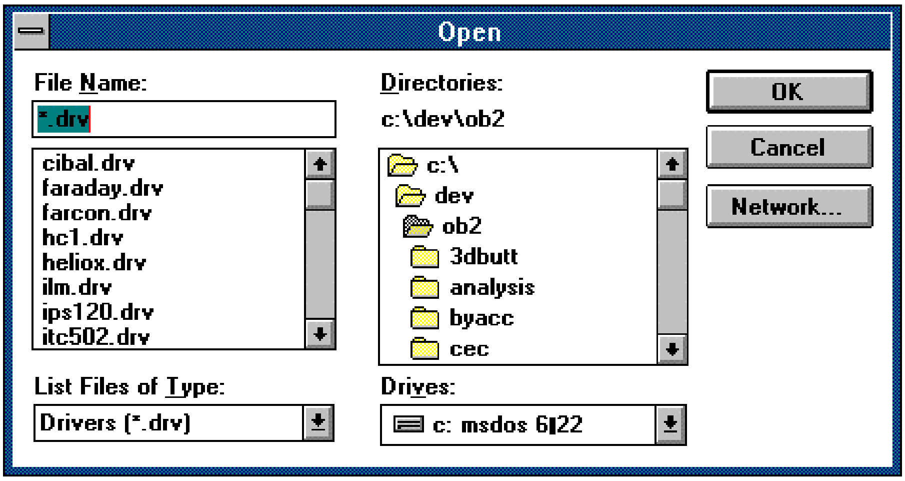


Driver: Configuration:

ITC503: default

Figure 7 The Configure Instruments Dialog showing Instrument Drivers in use.

To add further instruments, click on the "add" button. A list of available instrument drivers will be displayed: note that this list may vary from system to system, depending on the drivers actually supplied.



File Name: *.drv

cibal.drv
faraday.drv
farcon.drv
hc1.drv
heliox.drv
ilm.drv
ips120.drv
itc502.drv

Directories: c:\dev\ob2

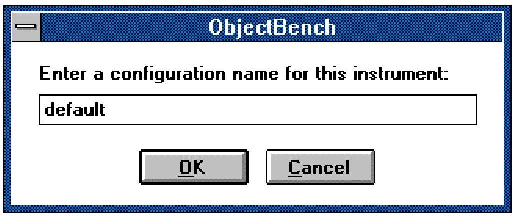
c:\
dev
ob2
3dbutt
analysis
byacc
cec

List Files of Type: Drivers (*.drv)

Drives: c: msdos 6|22

Figure 8 The Add Instruments dialog box, showing a list of instrument drivers available for loading.

Figure 8 shows a typical list of instruments. To load an instrument driver, select one from the list using the mouse and click on "OK". Another dialog will appear, prompting you to enter a configuration name for the instrument as shown in Figure 9.



Enter a configuration name for this instrument:

default

Figure 9 - The Instrument Configuration Name Dialog

The configuration should consist of up to 8 characters which are used to distinguish multiple instances of the same instrument driver. In this example, instrument is about to be loaded with a configuration name of "default" - further instances of the same instrument could be loaded subsequently with different configuration names. The configuration is used when creating a file name for storing the instrument's setup. In this example, setup information about an ITC503 would be saved in a file called "default.503" when exiting from ObjectBench, and retrieved from this file when ObjectBench is next run.

4.2.2 Controlling an Instrument from ObjectBench

Once an instrument has been loaded it can be controlled and its signals read in several ways. The simplest method is to control the instrument from its software front panel, which depends on the specific instrument. This facility will be described in later chapters of this manual. Another method is to control the instrument via the built in macro language, which again will be described in a later chapter. Finally, instrument signals may be displayed graphically as described in the next section.

4.3 How to Plot Graphs of Instrument Signals

When one or more instruments have been loaded as described in previous sections, it is possible to plot graphs of signals available from the instruments (for example, temperature or field strength). Before attempting this, refer to later sections of this manual for information on how to control your particular instruments, and "connect" them so that they are obtaining measurements in real time.

Any number of different XY plot windows may be opened and in operation at any time. However, there is an upper limit of 1000 data points allowed in any one curve. After this limit has been exceeded, early data points will be gradually discarded as new data is added. This is done to keep the display speed at an acceptable level. If it is necessary to record more than 1000 data points, use a log file to store all the data as described in section 4.3.3. This file may subsequently be loaded into an analysis window which is capable of displaying any number of points.

To create a new XY plot window, press or select the "XY Plot" option from the "Windows" main menu option. You may close existing windows by double clicking the top left corner of the window.

The appearance of the graph can be changed at any time as described in 4.4.

4.3.1 Selecting the Signals to Plot

The XY plot window is a window containing a graph. It may be moved or resized to any position on the screen. Initially an empty graph is displayed, ready to receive data as shown in Figure 10.

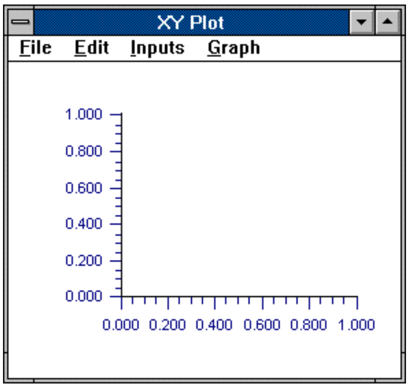
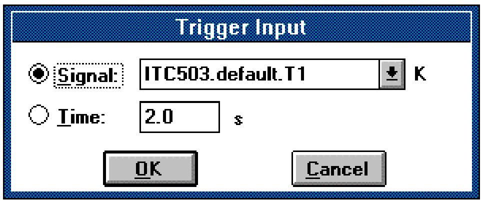


Figure 10 An empty XY Plot window.

The most important menu option is "Inputs", because this is used to select which signals are to be plotted, and when. There are three decisions to make: the choice of X parameter; the choice(s) of Y parameter(s); and the trigger.

The trigger is the event that causes a point to be plotted on the graph - there are two options. Data can be read and plotted at regular time intervals, or it can be plotted every time an instrument driver makes a new signal value available.


Trigger Input

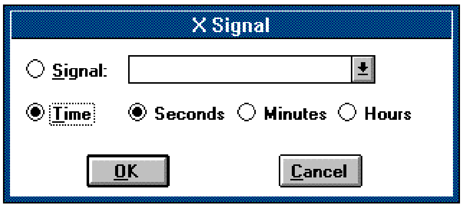
● Signal: ITC503.default.T1 K

○ Time: 2.0 s

Figure 11 Selecting the trigger for an XY plot.

Figure 11 shows the dialog which is displayed when the Inputs/Trigger menu option is selected. The choice of signal or time trigger may be made by clicking one the radio buttons on the left of the dialog box. If signal triggering is selected, a signal must be selected from drop-down list. In this example, signal itc503.default.T1 has been selected. Click on "OK" to confirm this choice.

Having chosen the trigger source, you may now select the signal to be plotted on the X axis. Select the "Inputs"/"X signal" menu option.


X Signal

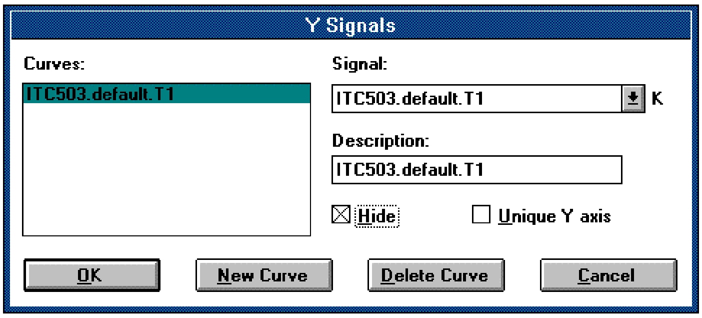
○ Signal:

● Time ● Seconds ○ Minutes ○ Hours

Figure 12 Choice of the signal to display on the X axis of a plot.

Figure 12 shows the dialog box used to select the x parameter. There is a choice between using elapsed time or an instrument signal. As before, click on a radio button to make your choice, and select a signal from the drop-down list if necessary. In this example, time has been chosen for plotting on the X axis, with units of seconds.

Finally, choose the signals to be plotted on the Y axis. Select the Inputs/Y signals option to display the dialog box shown in Figure 13.



Curves:
ITC503.default.T1
Signal: ITC503.default.T1 K
Description: ITC503.default.T1
☒ Hide
□ Unique Y axis

Figure 13 The dialog box used to select Y signals in an XY plot.

This dialog allows one or more signals to be selected for use as Y coordinates of plotted points. To create a new curve (that is a plot of Y vs. X) click the "New Curve" button with the mouse. This will cause a new entry to appear in the list of curves with the text "new curve". Select this curve by clicking on it with the mouse, and then use the drop-down list of signals (to the right) to select which signal should be used. When a signal has been selected, the entry in the list of curves will change to a description of the curve, in this example, "itc503.default.T1 vs. time". If you wish, you may change the description by editing it in the edit box labelled "Description" (to the right). This description will appear in the graph key. When you have created all the curves you require, click on the "OK" button.

There are two check boxes which provide access to two further features in this dialog box. Selecting "Hide" will make the selected curve invisible, without deleting its data. It may later be displayed by clicking on "Hide" again. This feature can be useful if you wish to omit one or more curves when the graph is printed.

Selecting "Unique Y axis" forces the graph to include a separate Y axis for this curve. Normally, the software attempts to share Y axes between curves if the points plotted have the same units, creating one Y axis for each set of units (for example field and temperature). When "Unique Y axis" is selected this feature is overridden.

4.3.2 "Connecting" the XY Plot

Once the signals to be plotted have been selected as described in the previous section, select the "Signals"/"Connect" menu option to start plotting data. The relevant instruments must also be connected before data can be acquired. You can press to connect all the graphs and all the instruments, or use the "System"/"Connect all" menu option.

While the plot is connected, it is no longer possible to change the signals chosen for plotting.

To disconnect the XY plot, select the "Signals"/"Connect" option again, or the main menu option "System"/"Disconnect all". You can disconnect all XY plots and all instruments by pressing .

Sometimes it is useful to clear all old data from the graph before acquiring new data. This may be accomplished using the "Graph"/"Clear data" menu option.

4.3.3 Saving data to a file

You may store the data points recorded in an XY graph in a data file. There are two ways to save the data.

The first is a simple save of the graph displayed, which is performed by using the XY graph's "File"/"Save" menu option. You are prompted for a file name and the data is saved in the file. The data may be viewed at a later date by opening an analysis window and loading the data file. Note that this method of saving data saves exactly the data visible in the graph, so if the limit of 1000 points plotted in the graph has been exceeded, and old data lost, the old data is not saved to the file either. To be sure of not losing old data, you should instead save data to a file using the logging option.

When logging data to file, each data point is written to the file as it is read from hardware and plotted on the graph. File data is never discarded. Be aware that it is possible to create very large files in this way. To log data to a file, use the "File"/"Log" menu options. You will be prompted for a file name, and then logging will commence. To check that data is being logging, look at the "File" menu. You should see a small tick beside the "Log" option. To stop logging, select this option again. The tick will disappear.

4.4 Graph Operations

ObjectBench displays data in graphical format in XY plots, in analysis windows and in graph windows generated by the macro language. In each case a set of operations performed on the graph, such as rescaling. These operations are described in this section of the manual.

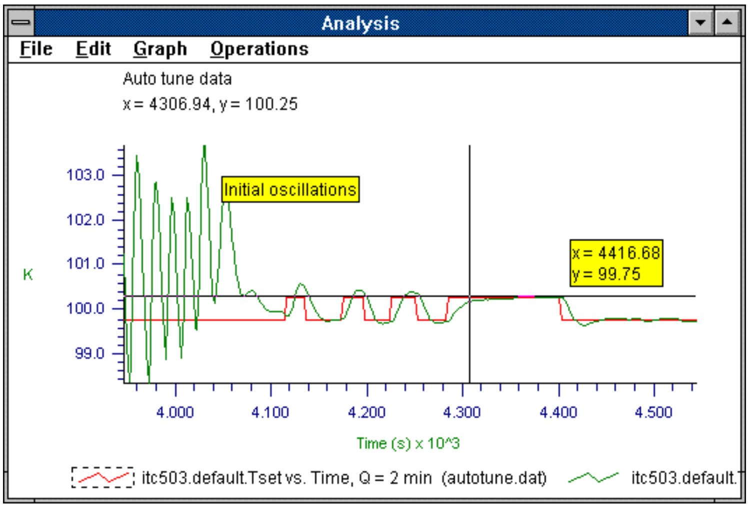


Figure 14 - A Typical Graph.

A typical graph is shown in Figure 14. A key appears beneath the graph which contains an entry for every curve in the graph; the key for the currently selected curve is surrounded by a dashed rectangle as shown. There is a cross (the readout cursor) on the graph which applies to the currently selected curve. The coordinates of the cross hair appear above the graph. Above the cross hair coordinates there is some descriptive text, in this case “Auto tune data”.

The graph can be manipulated using the mouse, the main menu, and a popup menu accessed through the right mouse button. All these features are described in the following sections.

4.4.1 Mouse Operations

As the mouse pointer is moved around over the graph window, it changes from the usual Windows mouse pointer when it has a special function. Several features require you to drag the mouse. The mouse is dragged by pressing the left mouse button at the starting position, moving the mouse to the finishing position, and releasing the button. The mouse is double clicked by rapidly pressing and releasing the left mouse button twice. A popup menu, the graph popup menu, is accessed using the mouse as described in the table. Popup menus are used most easily by pressing the right mouse button to display the menu, moving the mouse to the menu option you require while still holding the button down, and finally releasing the button to select the option. To select no option, release the right button when the mouse pointer is outside the menu. The menu options are described in subsequent sections of this manual.

Here are the different possibilities:

+↘	The mouse pointer is over a graph and can be used to rescale the graph by zooming in on a selected rectangular region. Drag the mouse to select a rectangular outline on the graph. Clicking with the left mouse button places the readout cursor at the position on its curve nearest to the position where the mouse was clicked. Clicking with the right mouse button displays the graph popup menu (see section 4.4.5).
☝	The mouse pointer is over the key for a particular curve. Clicking with left mouse button on a curve’s key selects that curve. If the curve is already selected, all curves are deselected. Clicking with the right mouse button displays the curve popup menu (see section 4.4.6). Double clicking on the descriptive text allows the text to be edited.
⇐|⇒	The mouse pointer is over the readout cursor, and can be used to drag the cursor along its curve. The readout cursor is always on the currently selected curve.
✋	The mouse pointer is over an object on the graph such as a text label. The mouse can be used to drag the object to a new position on the graph. Double clicking the mouse displays a dialog which allows the object to be edited.

Table 1 - Mouse Pointers and their Functions

4.4.2 Graph Main Menu Options

Many graph functions are controlled from the main menu option “Graph” shown in Figure 15.

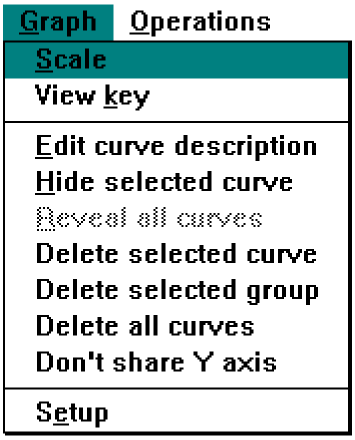


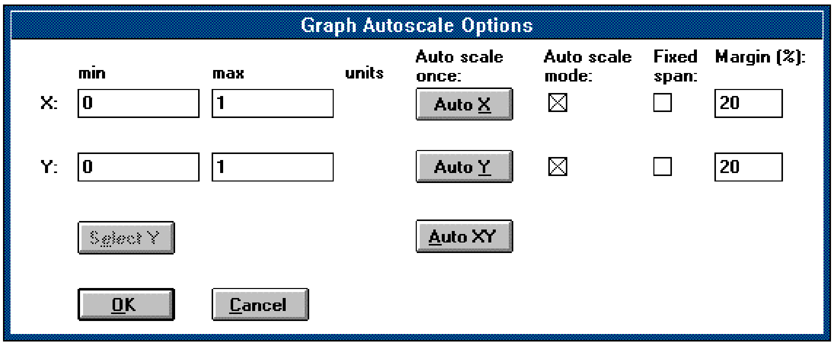
Figure 15 - The Graph Main Menu

Each option is described below. Options which are not accessible at any time are displayed in grey instead of black.

Scale	The scaling dialog is displayed allow axis ranges to be set manually, and autoranging to be configured. See section 4.4.3.
View key	A curve key is displayed in a scrolling window. This is an alternative to the curve key displayed below the graph. It is most useful when there are a large number of curves in the graph. It is not printed with the graph.
Edit curve description	See the Curve Popup Menu , section 4.4.6.
Hide selected curve	See the Curve Popup Menu , section 4.4.6.
Delete selected curve	See the Curve Popup Menu , section 4.4.6.
Delete selected group	See the Curve Popup Menu , section 4.4.6.
Delete all curves	See the Curve Popup Menu , section 4.4.6.
Don’t share Y axis	See the Curve Popup Menu , section 4.4.6.
Setup	The graph setup dialog is displayed - see section 4.4.4.

4.4.3 The Graph Scaling Dialog

The graph scaling dialog is accessed from the "Graph"/"Scale" main menu option. It is shown in Figure 16.



	min	max	units	Auto scale once:	Auto scale mode:	Fixed span:	Margin (%):
X:	0	1		Auto X	☒	☐	20
Y:	0	1		Auto Y	☒	☐	20

Figure 16 - The Graph Scaling Dialog

The ranges of the X and Y axes can be entered as required. If there is more than one Y axis, click on "Select Y" to cycle through them.

You can automatically adjust the range of the X or Y axis to match the range of the data (autoscale) actually present by pressing the "Auto X" or "Auto Y" button respectively. Both X and Y ranges can be autoscaled by pressing the "Auto XY" button.

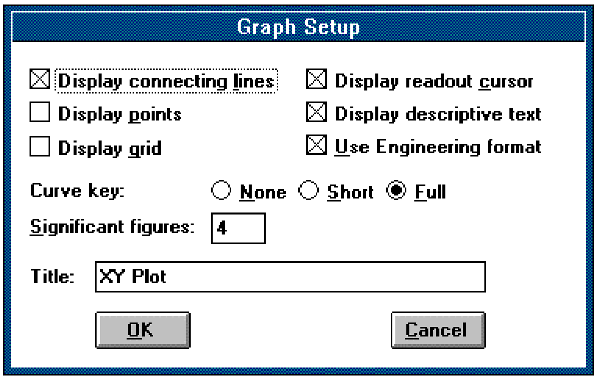
This dialog also allows autoscaling to be performed automatically when new data is added to the graph. Put a tick by "auto scale mode" for X or Y if you want that axis to adjust automatically to accommodate new data points as data is added to the graph. In this mode, an axis range is extended beyond the value of the new data point by a fixed percentage, when can be entered in the "Margin" field for that axis. The margin is used to avoid rapidly repeated rescaling of a graph as each new data point is added.

It is possible for the span of an axis to remain unchanged when it autoscales automatically as new data is added. The axis range is offset instead of extended. Place a tick beneath "Fixed span" if you want to use this feature.

4.4.4 The Graph Setup Dialog

The graph setup dialog is accessed from the "Graph"/"Setup" main menu option. It is shown in Figure 17.

You can choose whether to display optional elements of the graph by placing ticks in the dialog. The connecting lines between data points, the cross at the data point, the axis grid and the readout cursor are all optional. The descriptive text, which is not always present, above the graph can also be optionally displayed. If engineering format is selected, all axis exponents are multiples of 3.



Graph Setup

☒ Display connecting lines

☐ Display points

☐ Display grid

☒ Display readout cursor

☒ Display descriptive text

☒ Use Engineering format

Curve key: ○ None ○ Short ● Full

Significant figures: 4

Title: XY Plot

Figure 17 - The Graph Setup Dialog

The curve key can appear below the graph. It displays a sample of each curve's line colour, with some descriptive text. If a short key is selected, a number is displayed instead of the descriptive text.

The number of significant figures to display in axis labels can be entered in the "Significant figures" field.

If a title is entered, it appears at the top of any graphs printed out.

4.4.5 The Graph Popup Menu

If the right mouse button is pressed when the mouse pointer is over the graph, a popup menu appears as shown in Figure 18.

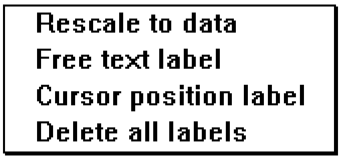


Figure 18 - The Graph Popup Menu

Menu options which are not currently available are displayed in grey. Each option is described below.

Rescale to data	Rescale the X and Y axes to match the range of the data displayed exactly.
Free text label	The readout cursor must be moved to a point on a curve before this option is available. This option allows a box containing text to be place on the graph. A dialog box invites you to enter any text you wish. The box is then placed on the graph, and may by positioned by dragging it with the mouse. Note that the text label is always positioned relative to the curve point that was selected when you created it. When the graph's scale is changed and the point's position on the display changes, the text label will also change position. A text label may be moved at any time by dragging it with the mouse. Double click on the label to display an editing dialog which allows you to change the text, change the label's colour, or delete the label.
Cursor position label	A cursor position label is exactly like a free text label (see above), except that it initially contains the x and y coordinates of the readout cursor.
Delete all labels	All text and cursor position labels are removed from the graph.

4.4.6 The Curve Popup Menu

If the right mouse button is pressed when the mouse pointer is over a curve key, a popup menu appears as shown in Figure 19. Note that the curve key is only visible when there is at least one curve in the graph and the curve key is not set to "none" in the graph setup dialog (see a later section in this manual).

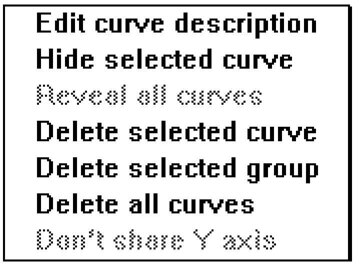


Figure 19 - The Curve Popup Menu

Menu options which are not currently available are displayed in grey. Each option is described below.

Edit curve description	A dialog appears which allows a small amount of text to be entered for the curve key.
Hide selected curve	The currently selected curve and its key are hidden. All hidden curves may be revealed by selecting "Reveal all curves" from this menu.
Delete selected curve	The selected curve is deleted.
Delete selected group	All curves in the same group as the selected curve are deleted. A group of curves all share the same x data and can be loaded from and saved to a single data file.
Delete all curves	All curves in the graph are deleted.
Don't share Y axis	Whenever possible, curves with the same Y units share the same Y axis. This may be overridden by selecting this menu option.

4.4.7 Copy a Graph to the Windows Clipboard

The contents of a graph may be copied to the Windows clipboard by using the "Edit"/"Copy" menu option. This allows graphs to be pasted into other Windows programs such as Microsoft Word, for preparation of reports. Graphs are copied to the clipboard in bitmap format.

4.4.8 Printing and Plotting

A graph may be printed or plotted to any printer or plotter supported by Windows. To print a graph, first select the "File"/"Printer setup" menu option. A list of printers available on your system will be displayed. Select the one you require. You may configure the printer settings, if you wish, by clicking on the Setup button. Otherwise, click on OK. Now you may print the graph by using the "File"/"Print" menu option. A dialog box will be displayed briefly as data is transferred internally to the Windows print spooler, and then printing will commence. Note that printing at high dot resolutions on laser and matrix printers can take some time, so it is often better to choose a medium or low resolution using the printer setup button described above.

If you wish, you can preview the printout by using the "File"/"Print preview" menu option.

4.4.9 Graph Fonts and Colours

All graphs in ObjectBench share the same colours and use the same fonts for text. These colours and fonts can be changed by using the main menu items "Configure"/"Graph colours" and "Configure"/"Graph fonts". The colour and font used for each part of the graph can be chosen. These settings will be retained for future use by ObjectBench.

4.5 How to use ObjectBench Macro Commands

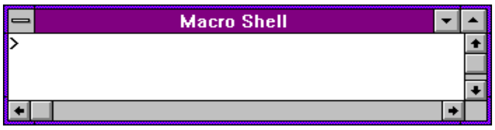
ObjectBench includes a macro interpreter that can be used to control instruments and acquire data from them in a very flexible way. Macros may be entered into the macro interpreter for execution at once in immediate mode. Alternatively, several macros may be written in a text file using an editor, for execution as a program of operations. Such a macro file may be invoked from a custom dialog box created using Borland's Resource Workshop.

When creating a macro file, any ASCII editor may be used. ObjectBench includes a suitable editor which may be invoked using the "Windows"/"Edit macro" menu options. Any number of such editor windows may be opened and used.

The following sections describe how to execute individual macros commands and how to create a macro file. Later chapters provide detailed reference information on the macro language and describe how to call macro files from custom dialog boxes.

4.5.1 Executing Macro Commands in Immediate Mode

To execute macro commands in immediate mode, create a Macro Shell window using the "Windows"/"Macro" shell menu options. A window similar to Figure 20 will be created, with ">" prompt inviting a command to be entered.

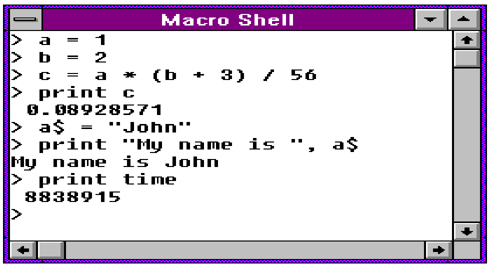


```
>
```

Figure 20 A macro shell Window.

The rest of this section provides an informal introduction to immediate mode macro commands. Full reference information on macro commands is included in section 7.

The macro command shell is an interpreter which translates commands before executing them, to achieve fast execution time. It is similar in many ways to BASIC, so that, for example, variables may be used in arithmetic expressions and printed. Figure 21 shows examples of how to use ObjectBench variables.



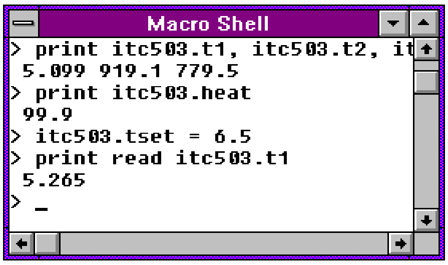
```
> a = 1
> b = 2
> c = a * (b + 3) / 56
> print c
 0.08928571
> a$ = "John"
> print "My name is ", a$
My name is John
> print time
 8838915
>
```

Figure 21 Typical macro operations.

Upper and lower case variables are equivalent and all variables are double precision floating point. Variable names may be of any length.

In this example, a print statement is used to print the value of a variable. The same example shows how string variables may be assigned values and printed in a similar way. There is also a special variable named "time" whose value is the time elapsed in milliseconds since Windows was started.

The macro command language supports a special type of variable for accessing instrument signals. For example, if there is an ITC503 present and connected in the system, its signals might be accessed as shown in Figure 22.



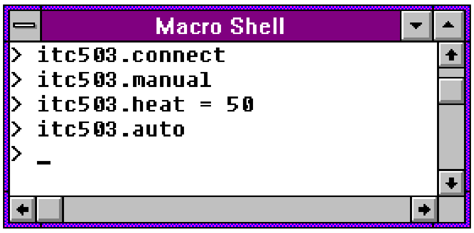
```
> print itc503.t1, itc503.t2, it
 5.099 919.1 779.5
> print itc503.heat
 99.9
> itc503.tset = 6.5
> print read itc503.t1
 5.265
> _
```

Figure 22 Accessing instrument signals from the macro shell.

This example shows how to print the three channel temperatures and the heater output of the ITC503. It also shows how to set the set point for automatic temperature control. Details of the ITC503 macro commands may be found in the appropriate chapter in this manual if an ITC503 is included in your system.

Note that the values printed are those read most recently by the instrument driver. However, these readings may be out of date if the instrument's update period is set to a large value. A fresh up-to-date reading may be obtained by inserting the "read" keyword as shown in the last command in the example (Figure 22).

Specific instrument commands may be executed as shown in Figure 23.



```
> itc503.connect
> itc503.manual
> itc503.heat = 50
> itc503.auto
> _
```

Figure 23 Example showing how to issue instrument commands in the macro shell.

In this example, the ITC503 is connected, set to manual mode, its heater output is set to 50%, and finally it is set to auto control mode. Available commands vary from instrument to instrument, and are listed for instruments in your system in later sections of this manual.

4.5.2 Creating and Executing Macro Programs

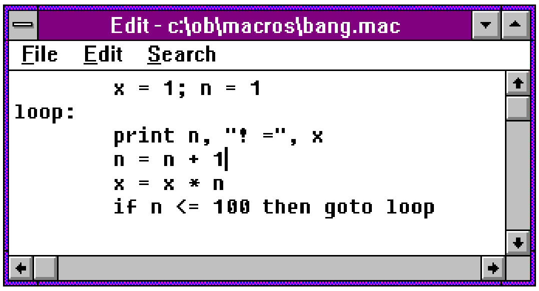
Macro programs are simply files containing sequences of macro commands. In addition to those macro commands described in the previous section, there are flow control and conditional commands similar to those in BASIC which make it easy to write powerful macro programs.

This section provides an informal introduction to the extra macro commands used in macro files. Full reference information on macro commands is included in section 7.

The macro program shown in Figure 24 calculates factorials of numbers in the range 1 to 100. It illustrates how to use labels and conditional jumps to control program execution. As well as **goto** *label*, ObjectBench macro language supports **gosub** *label* with **return** as in BASIC. Conditional statements can make jumps depending on the outcome of a test as shown. A label consists of any text followed by a colon. Program execution finishes if a **stop** statement or the end of a macro file is encountered.

To test this example, type the commands shown into a file using a file window, and save it in the default macro directory as "bang.mac". The default macro directory is specified in the main menu option "Setup/Directories" and then go to a macro shell window. Type the following to run the macro:

load "bang.mac" + <ENTER>
run + <ENTER>



```
        x = 1; n = 1
loop:
        print n, "! =", x
        n = n + 1
        x = x * n
        if n <= 100 then goto loop
```

Figure 24 A macro program for calculating factorials.

At this point, a list of factorials will be displayed. If you wish to interrupt a macro before it has finished, press the <ESCAPE> key at any time. If you want to respond to the <ESCAPE> key in a special way in a macro program, define a label "escape:". When <ESCAPE> is pressed, program execution transfers to this label. Pressing <ESCAPE> a second time always interrupts a macro program.

A macro may be interrupted by pressing <SPACE> at any time during execution. Press <SPACE> again to resume execution, or <ESCAPE> to interrupt execution.

Note that when a macro has been loaded into a macro shell, ObjectBench remembers this and automatically loads the same macro next time ObjectBench is run. There is also an option in the macro shell's system menu, "auto run", which can be selected to execute the macro when it is loaded automatically.

Data files may be created, written to, and closed from macros. The mechanism for doing this differs from that in BASIC, to make sure that all data files are in a compatible format that may be read by an analysis window. A typical example of creating, writing to and closing a data file is as follows:

```
format #1, x, y
open #1, file = "test.dat", "This is a description"
x = 1; y = 2
write #1
close #1
```

This example first declares a format, which comprises two columns, one for X and one for Y. "#1" is the channel identifier. The second line opens a data file called test.dat with the descriptive text shown. To write to the file, the simple command shown on the fourth line is used, and finally the file is closed. It is possible to specify column labels and units for the data file: this is described more fully under **format** in the macro command reference section. Any number of independent data files may be manipulated by using different channel identifier values. If the file name contains a wildcard character (“*” or “?”) a standard Windows file selection dialog is displayed.

Data may be transferred to the screen or a graph in a very similar way to writing it to a file. To print the values on the screen, replace the **open** command in the example above with the following:

```
open #1, graph = "The graph title"
```

This command will create a new graph window ready to receive data. Similarly, substituting:

```
open #1, screen
```

will direct output to the macro shell window. This feature is useful for debugging.

Note that the same channel may be opened for any combination of file, graph or screen output at the same time by simply including more than one **open** command in the macro.

The usual way of acquiring data from an instrument is to assign values from the special instrument variables described in the previous section: this is known as synchronous data acquisition. Sometimes, however, it is desirable to respond to an instrument measurement event: this is known is *asynchronous* data acquisition. This may be implemented in the ObjectBench macro language using **the on signal xxxx / return** commands. The following example shows how to respond to measurements of a signal **lockin.X**:

```
rem The main program:
on signal on
loop:
goto loop
rem The signal measurement subroutine:
on signal lockin.x
print "lockin.x = ", lockin.x
return
```

This program consists of two parts. The first part is just an infinite loop that does nothing. However, whenever instrument **lockin** reads a new value for **x**, the program jumps to the statement after **on signal lockin.x**. In this example, the value is printed and the return statement causes the main program loop to be continued. Notice the command **on signal on** near the start of the program, which activates the on signal mechanism. Until this command is executed, the *on signal handler* is not invoked. Several *on signal handlers* may be defined for a number of different signals.

Sometimes it is useful to include common definitions or subroutines in a macro program. This may be achieved in the ObjectBench macro language by using the **include** statement. This reads a file into the program so that the contents of the file appear at that point in the program.

The complete example macro program shown in Figure 27 illustrates many of the concepts discussed in this section. It is the macro used to measure sample moment versus time in a vibrating sample magnetometer (VSM) system experiment. It is normally invoked from a custom dialog (described elsewhere in this manual) which loads the macro program, sets the values of some variables, and runs it.

The example first declares some formats for file and graph data output, before opening a data file and graph to receive output. Then it enables *on signal* handling and waits for a defined time before closing the data output channels and finishing. The subroutine *wait* is defined in the included file "util.mac" on the last line of the program. If escape is pressed while the program is running, execution transfers to the "escape:" label.

An *on signal* handler is included which is activated each time the signal vsm.m is measured. This subroutine measures the elapsed time by calling another subroutine in "util.mac", and then writes data to the file and graph before returning.

4.6 How to connect Signals using Signal Links

Sometimes it is useful to be able to update one instrument input signal from another instrument output signal. For example, a Vibrating Sample Magnetometer (VSM) instrument has a signal input named "V" which has to be updated from the measured hardware voltage signal. This signal is available from a lockin amplifier channel (X or Y) which has corresponding output signals named "X" and "Y".

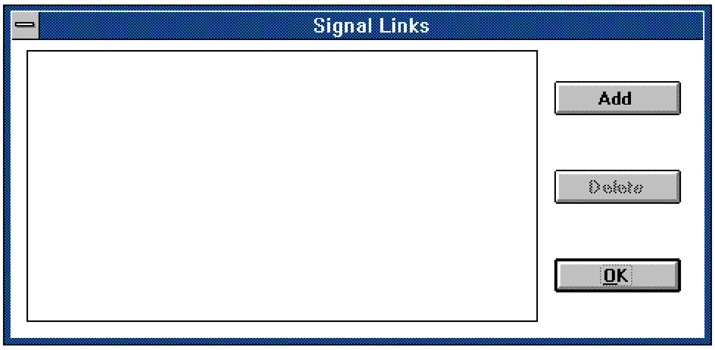
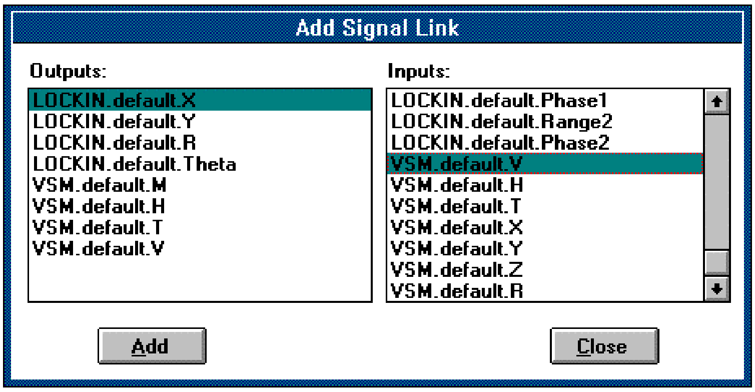


Figure 25 The empty signal links dialog.

This will display a list of any existing signal links, as shown in Figure 25. Initially the list will be empty.

To add a new signal link, click on the Add button. A dialog similar to that shown in Figure 26 will be displayed.



Outputs:

LOCKIN.default.X
LOCKIN.default.Y
LOCKIN.default.R
LOCKIN.default.Theta
VSM.default.M
VSM.default.H
VSM.default.T
VSM.default.V

Inputs:

LOCKIN.default.Phase1
LOCKIN.default.Range2
LOCKIN.default.Phase2
VSM.default.V
VSM.default.H
VSM.default.T
VSM.default.X
VSM.default.Y
VSM.default.Z
VSM.default.R

Figure 26 A typical Add Signal Links dialog.

The Add Signal Link dialog contains two scrolling lists of signals. In the left list appear all the output signals available from all the instruments in the system, while the right list contains all the input signals. To create a signal link, select one signal from each list by clicking on them with the mouse, and click on the Add button. Continue in this way until all required signal links have been created, and finally click on the Close button.

In this example, the lockin amplifier X output is linked to the vsm instrument V input. Every time a new value for lockin.X is available, it is written to the vsm.V signal.

Note that only signals with the update characteristic appear in the list of outputs, while only signals with the write characteristic appear in the list of inputs. See section 7.2.

```
        rem Moment v Time
        rem ~~~~~~~~~~~~~
        rem The following parameters must be set up:
        rem momtfilename$
        rem runtime
        rem momtdesc$
        rem Some general initialisations:
        include "init.mac"
        format #1, elapsed["t","s"], vsm.m["moment", "emu"],    vsm.V["Signal","V"],
vsm.t["temp","K"], vsm.h["Field", "T"]
        format #2, elapsed["t","s"], vsm.m["moment", "emu"]
                open #1, file = momtfilename$, momtdesc$, overwrite
                open #2, graph= "Moment vs Time", overwrite
        rem Collect the data:
                on signal on
                waitdwell = runtime * 1000
                gosub wait
        escape:
        rem Close the data file:
                on signal off
                close #1
                close #2
                print "Moment vs. Time macro finished."
                stop
        on signal vsm.m
                gosub elapsed
                write #1
                write #2
                print elapsed, " of ", runtime, " seconds elapsed."
                return
        rem Include some utility subroutines:
        include "util.mac"
```

Figure 27 An example ObjectBench macro program.

To connect two signals, select the "Configure"/"Signal Links" option from the main ObjectBench menu.

4.7 How to Analyse Data Files

Data files can be created and filled with data in a number of ways, such as creating an XY plotter log file, or creating an output file channel in a macro program. All these files are in the same format and can be read into an ObjectBench analysis window. To create an analysis window press or select the "Windows"/"Graphical Analysis" option from the main ObjectBench menu. A typical analysis window, containing data, is show in Figure 28.

Analysis Windows contain graphs which behave as described in section 4.4.

Analysis windows allow data files to be read and created, and some simple operations performed on the data. These steps are described following sections.

4.7.1 Loading Data from Data Files

To load data from an ObjectBench format data file, select the "File"/"Open" option from the analysis window's menu. A standard open file dialog will be displayed, allowing you to select your data file from a list as shown in Figure 30. The example shows the selection of a file named "resp3.dat".

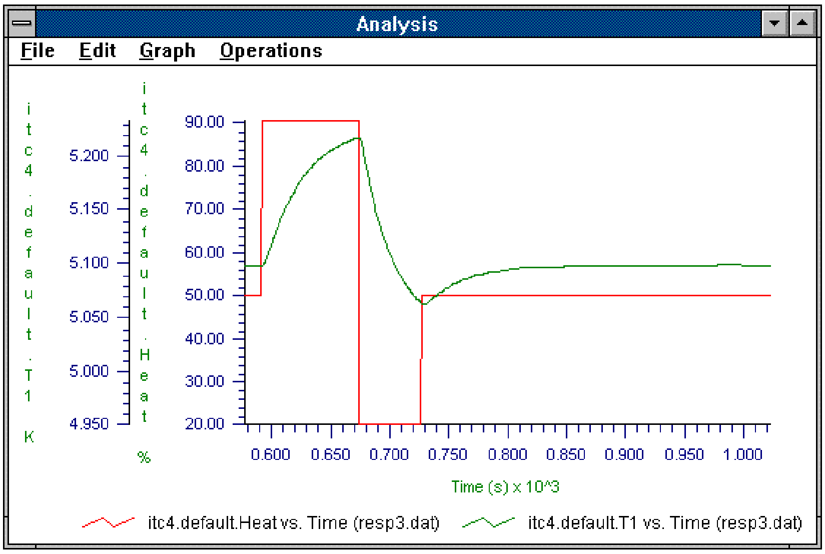
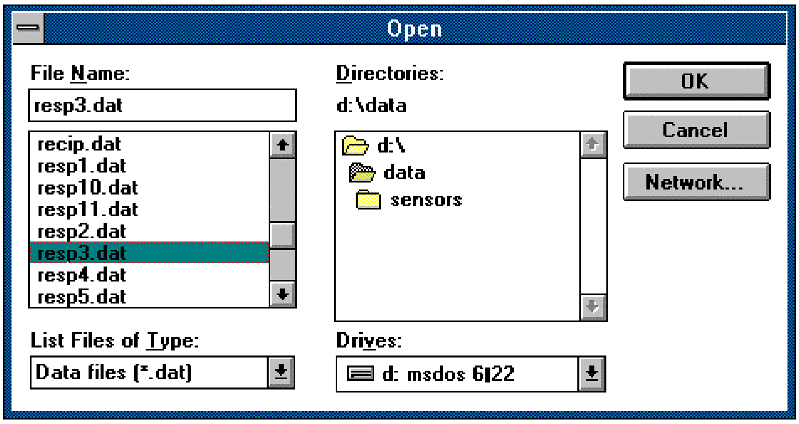


Figure 28 A typical analysis window. In this example, the temperature response of system to a stepped heater power is shown.

The load file curves dialog displays a list of all the signals stored in the file, one entry for each column of data. At the top of the dialog is a drop-down list which allows you to choose which signal to use for the X axis. The remaining signals are displayed in a scrolling list (labelled "Y data") and initially all are selected. An entry is selected when it is displayed in reverse text, as shown in the Figure. All selected signals will be used for Y data, so that one curve in the graph will be created for each Y signal. To toggle signal selection, click on the entry with the mouse. To select all the entries, or none of them, click on the Select all or Clear all buttons respectively. When your selection is complete, click on the OK button.



File Name: resp3.dat

recip.dat
resp1.dat
resp10.dat
resp11.dat
resp2.dat
resp3.dat
resp4.dat
resp5.dat

Directories: d:\data

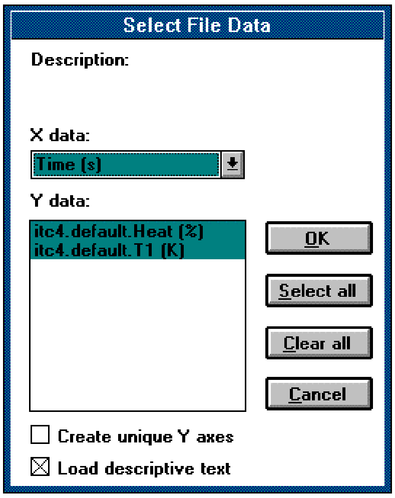
d:\
data
sensors

List Files of Type: Data files (*.dat)

Drives: d: msdos 6.22

Figure 29 Opening a file for the analysis window.

To load the file, click on the "OK" button. A second dialog similar to Figure 30 will be displayed.



Select File Data

Description:

X data:

Time (s)

Y data:

itc4.default.Heat (%)
itc4.default.T1 (K)

☐ Create unique Y axes

☒ Load descriptive text

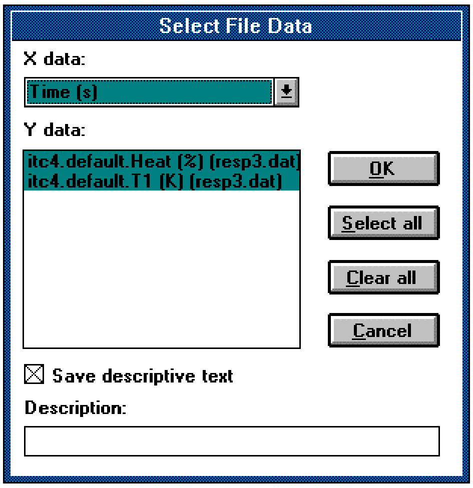
Figure 30 A load file dialog allowing selection of signals stored in the file.

Normally, the analysis graph will try to plot signals with the same units using the same Y axis range, thus using a minimum space on the screen for Y axes. If you wish to display all signals with separate Y axes, check the "create unique Y axes" box at the bottom of the dialog.

Notice that new curves from the data file are added to any previous curves existing on the graph. If you want to replace existing curves, use the "Graph"/"Clear" all data option before loading the data file. When adding curves to a graph which already contains data, the new X axis units must be the same as the existing X axis units.

4.7.2 Saving Graph Curves to a Data File

To save data from one or more curves to a data file, select the File/Save menu option. A dialog similar to that shown in Figure 31 will be displayed.



Select File Data

X data:

Time (s)

Y data:

itc4.default.Heat (%) (resp3.dat)
itc4.default.T1 (K) (resp3.dat)

☒ Save descriptive text

Description:

Figure 31 The Save File Curves dialog allowing selection of the data to save.

This dialog works in a very similar way to the Load File Curves dialog described above. You should select the X signal and Y signals that you wish to save in the data file. The only different feature in this dialog box is the "File description" field at the bottom. This offers an opportunity to enter descriptive text that will be saved in the file. The field scrolls horizontally so that any length of text may be entered.

The data chosen for the X axis in the drop-down list at the top of the dialog is that which will appear in the first column in the data file: it also determines which curves may be saved. Only those curves with chosen X data may be saved in this file.

When this dialog has been completed, click on OK button. A standard file dialog will appear which allows you to specify the file name for the data.

4.7.3 Performing Arithmetic between Curves

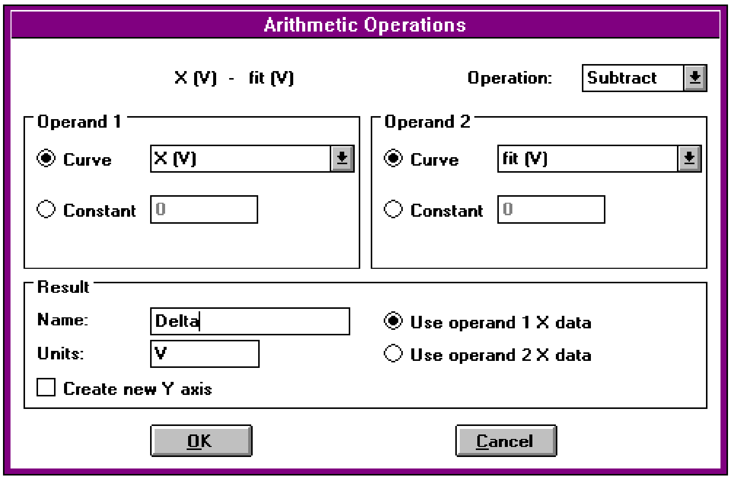
The "Operations"/"Arithmetic" menu option displays the curve arithmetic dialog shown in Figure 32. This dialog allows arithmetic operations (add, subtract, multiply or divide) to be performed between any two curves, or one curve and a constant value. If data points do not exactly correspond in X values between the two curves, the software uses linear interpolation on the data from one curve to generate the required data point. The software does not attempt to extrapolate data beyond the end of a curve.

Examples of when curve arithmetic is useful are for subtracting a background reading from a signal, or when it is necessary to normalise a signal to calibration data.

Displayed at the top of the dialog is the operation which is about to be carried out, in this example "X(V) - fit(V)". The operands are chosen from the two boxes labelled "Operand 1" and "Operand 2". These contain drop-down lists of available curves, and a field for entering a constant value. Click the radio button to indicate whether you wish to use the curve or the constant value. It is not permitted to choose a constant value for both operands. The operation that is to be performed is selected in a drop-down list in the top right of the dialog.

The box labelled "Result" determines how the resulting curve is to be created. The "Name" and "Units" fields allow you to enter the name and units of the resulting Y data.

Each data point in the resulting curve must have the same X value as a point from the existing curves. The two radio buttons to the right determine which X values to use. If "Use operand 1 X data" is selected, the new curve will share the X data for operand 1. X values for operand 2 will be interpolated as required. The resulting curve will share the X data for operand 1 and may thus be saved in the same data file (see section 4.7.2). Conversely, if "Use operand 2 X data" is selected, the new curve shares the X data for operand 2 and interpolates on operand 1, and may be saved in a file with operand 2.



Arithmetic Operations

X (V) - fit (V)

Operation: Subtract

Operand 1

● Curve X (V)

○ Constant 0

Operand 2

● Curve fit (V)

○ Constant 0

Result

Name: Delta

Units: V

□ Create new Y axis

● Use operand 1 X data

○ Use operand 2 X data

Figure 32 The curve arithmetic dialog, ready to subtract two curves.

When a division by zero would occur when dividing two curves, an invalid data point is generated and stored at that position, and the division of the remaining points continues as normal.

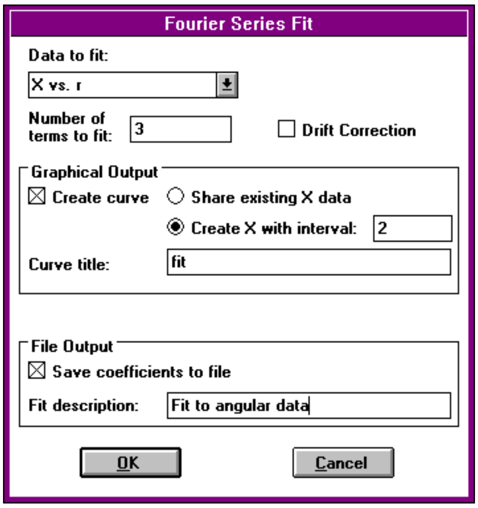
4.7.4 Fitting a Fourier Series to Periodic Data

Sometimes it is useful to be able to fit a Fourier series to periodic data, for example, when measuring the angular dependence of a parameter. A specific example is the measurement of a Vibrating Sample Magnetometer (VSM) moment dependency on angle. A harmonic fit to the data may be created and used for renormalising future readings.

To generate a Fourier series fit, select the analysis window menu option "Operations"/"Fit"/"Fourier". The dialog box shown in Figure 33 will be displayed.

There is a drop-down list at the top of the box which allows selection of the curve to fit. Below this, the number of Fourier series terms to include in the fit may be entered. Note that one term consists of two coefficients, one for the sine and one for the cosine component at the same frequency. The first term is the DC component. Thus, requesting three terms in the dialog would include the DC component and the first two harmonic amplitudes and phases. The wavelengths of the harmonics are integral divisions of the X axis range. Thus, if angular data is being fitted, it is important to be certain that the X axis range is exactly 360 degrees.

The "Drift Correction" check box allows a linear drift between the first and last point to be compensated. For example, if the signal at 0 degrees and 360 degrees differs, a linear correction to the signal can be applied which forces the first and last data points to have the same value.



Fourier Series Fit

Data to fit: X vs. r

Number of terms to fit: 3

☐ Drift Correction

Graphical Output

☒ Create curve

○ Share existing X data

● Create X with interval: 2

Curve title: fit

File Output

☒ Save coefficients to file

Fit description: Fit to angular data

Figure 33 The Fourier Fit dialog box.

The rest of the dialog pertains to the form of output from the fit. If graphical output of the fit function is required, check the "Create Curve" box and enter a title for the new curve. Use the radio buttons to determine whether the new curve will share the X data of the fitted curve, or will use new X data. The former option should be used if it is necessary to save the original and fit data in the same data file. The latter option is useful for interpolating between the original data points. If you choose to create new X data, indicate the new X interval you require.

If you wish to save the fit coefficients to a data file, check the "Save coefficients to file" box in the "File Output" box at the bottom of the dialog, and enter descriptive text for the file.

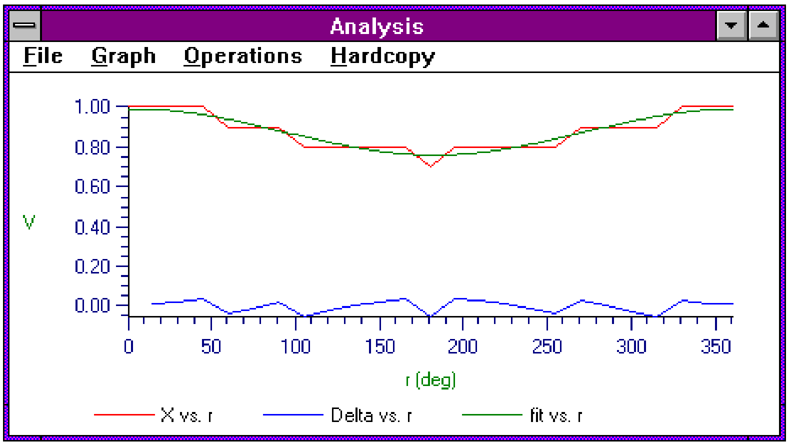


Figure 34 A typical analysis window, showing a Fourier fit to some data, and the residual.

To perform the fit, click on OK and wait for the calculations to be performed. If you chose to save the coefficients to a data file, a standard file dialog will appear for you to enter the file name.

Figure 34 shows a typical analysis window containing some angular data, a Fourier fit to that data, and the residual. The residual was created by subtracting the fit from the original data.

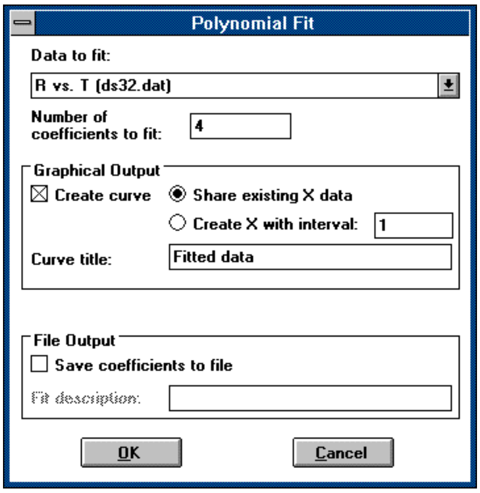
4.7.5 Fitting a Polynomial

To fit a polynomial expression to data, select the analysis window menu option "Operations"/"Fit"/"Polynomial". The dialog box shown in Figure 33 will be displayed.

There is a drop-down list at the top of the box which allows selection of the curve to fit. Below this, the number of polynomial terms to include in the fit may be entered. As an example, specifying 4 coefficients results in a third order fit.

The rest of the dialog pertains to the form of output from the fit. If graphical output of the fit function is required, check the "Create Curve" box and enter a title for the new curve. Use the radio buttons to determine whether the new curve will share the X data of the fitted curve, or will use new X data. The former option should be used if it is necessary to save the original and fit data in the same data file. The latter option is useful for interpolating between the original data points. If you choose to create new X data, indicate the new X interval you require.

If you wish to save the fit coefficients to a data file, check the "Save coefficients to file" box in the "File Output" box at the bottom of the dialog, and enter descriptive text for the file.



Polynomial Fit

Data to fit: R vs. T (ds32.dat)

Number of coefficients to fit: 4

Graphical Output

☒ Create curve ● Share existing X data

○ Create X with interval: 1

Curve title: Fitted data

File Output

☐ Save coefficients to file

Fit description:

Figure 35 The Polynomial Fit dialog box.

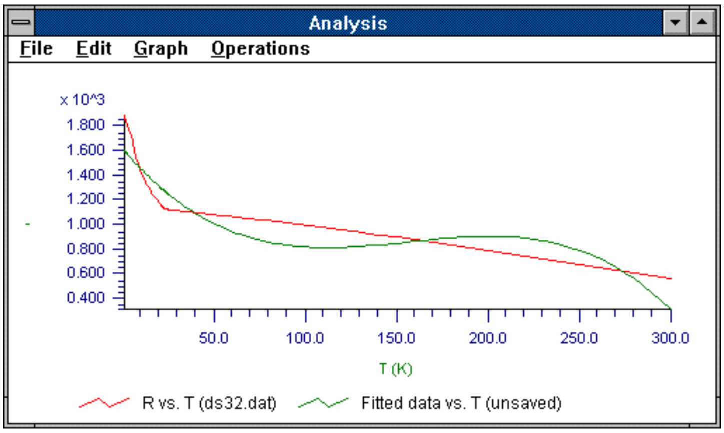


Figure 36 A typical analysis window, showing a third order polynomial fit to some data.

To perform the fit, click on OK and wait for the calculations to be performed. If you chose to save the coefficients to a data file, a standard file dialog will appear for you to enter the file name.

Figure 34 shows a typical analysis window containing some angular data, a Fourier fit to that data, and the residual. The residual was created by subtracting the fit from the original data.

4.8 Directories for Data Files

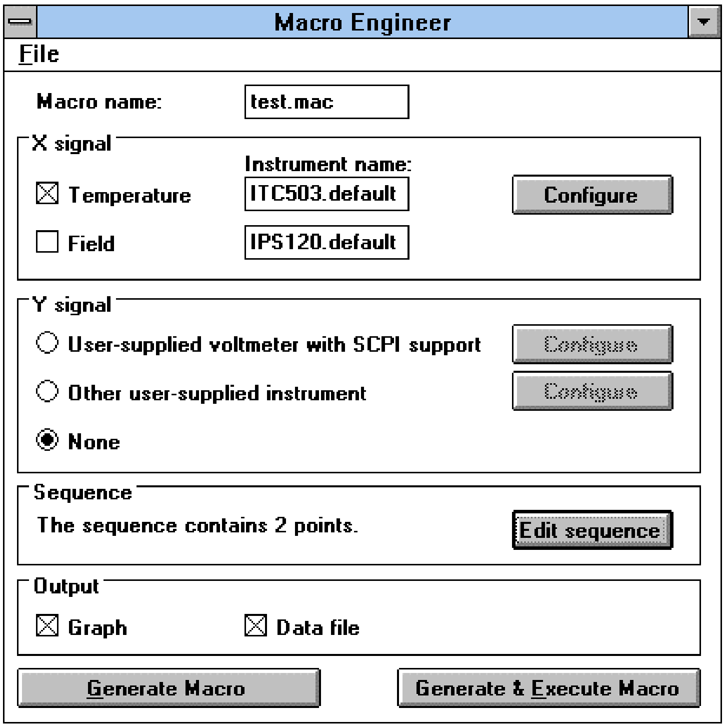
ObjectBench uses various types of file which are stored in default subdirectories of the ObjectBench program directory. The paths to these subdirectories may be changed by accessing the main menu "Configure"/"Directories" option. This displays a dialog box which allows you to enter the default paths for data files, macro files, and instrument drivers.

5 The Macro Engineer

The Macro Engineer is available only in versions of ObjectBench supplied with an Optistat system.

The Macro Engineer is used for generating macros automatically. It generates macros according to the user's specification for performing a sequence of temperature and field settings. At each temperature or field setting a reading may optionally be taken from an instrument supplied by the user. Alternatively, a macro supplied by the user may be called to make a measurement or perform any other task. Data acquired at each data point can be stored in a data file, or plotted in a graph.

You can access a Macro Engineer dialog by clicking on or selecting the main menu option "Window"/"Macro engineer" to display the dialog shown in Figure 37. You may open any number of these dialogs at once.



Macro name: test.mac

X signal

Instrument name:

☒ Temperature ITC503.default

☐ Field IPS120.default

Y signal

○ User-supplied voltmeter with SCPI support

○ Other user-supplied instrument

● None

Sequence

The sequence contains 2 points.

Output

☒ Graph ☒ Data file

Figure 37 - The Macro Engineer Dialog

First enter a name for the macro you wish to generate in the first field in the dialog. In this example, a macro named "test.mac" will be generated in the default macro directory.

The next section of the dialog allows you to select the X parameter, i.e. the independent control parameter for the sequence. This may be field or temperature or both. For each X parameter you select, you can specify the name of the instrument that is to be used for control. In most cases the default settings shown are sufficient. If you have more than one of a type of instrument type in your system, you will need to give them different names, and specify one of these names in this dialog. You can configure the temperature parameter as described in section 5.1.

The dialog section labelled “Y signal” allows you to control how the reading is taken at each temperature or field point. You may select “None” if you do not want any action to be taken at each temperature point. If you are taking a reading using a GPIB instrument with SCPI support, click on the appropriate option, and select configure to choose the type of reading you want. You can choose the reading type and range. Note that SCPI is not yet a rigid standard so you may have to modify and experiment with the macro that is generated.

If you want to make a measurement from a non-SCPI instrument or you want to perform some other action at each temperature or field point, you should select the “Other user-supplied instrument” option. You can then click on the “Configure” button to access the dialog described in section 5.2.

Click on the “Edit sequence” button to enter a sequence of temperatures or fields as described in section 5.3.

You can choose whether data is to be plotted on a graph or written to a file by selecting these options on the dialog.

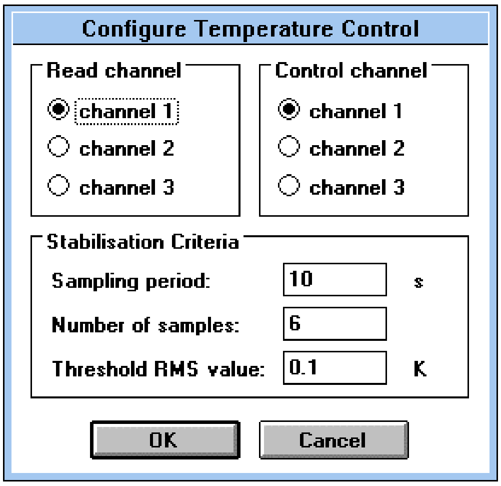
Finally, click on the “Generate macro” button to generate the macro. To run the macro, open a macro shell window and load and run the macro. Alternatively, click the “Generate and execute macro” button to perform all of these actions automatically.

You can save and load a set of Macro Engineer settings using the File menu in the Macro Engineer dialog.

5.1 Temperature Configuration Dialog

If you choose temperature as an X signal, you can click on the button labelled “Configure” to display the following configuration dialog:

The first two parts of the dialog allow you to select the control channel and reading channel for temperature control. Usually these will be the same channel. The third part allows you to specify temperature stabilisation criteria. These criteria are used by the macro to judge when temperature stability has been achieved at a requested temperature. The basic technique used is to maintain a running calculation of RMS deviation from mean of the temperature. Samples are taken at intervals of the sampling period supplied. When the specified number of samples has been taken, their RMS deviation is compared with the threshold value supplied. When it is less than or equal to this value, the temperature is accepted and the macro continues. If it is greater, temperature samples are taken until the deviation is less than the threshold.



Configure Temperature Control

Read channel

● channel 1
○ channel 2
○ channel 3

Control channel

● channel 1
○ channel 2
○ channel 3

Stabilisation Criteria

Sampling period: 10 s

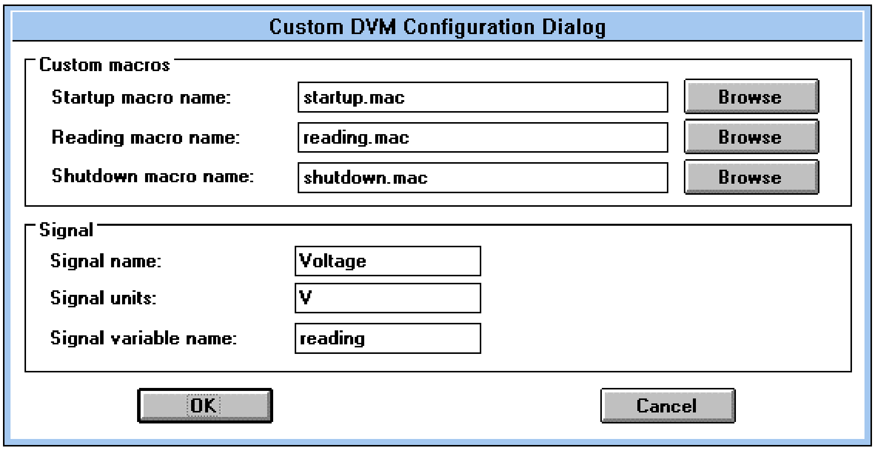
Number of samples: 6

Threshold RMS value: 0.1 K

Figure 38 - The Temperature Controller Configuration Dialog

5.2 Custom DMM Configuration Dialog

If you choose the "Other user-supplied instrument" option from the Macro Engineer dialog you can click on "Configure" to display the dialog shown in Figure 1.



Custom DVM Configuration Dialog

Custom macros

Startup macro name: startup.mac

Reading macro name: reading.mac

Shutdown macro name: shutdown.mac

Signal

Signal name: Voltage

Signal units: V

Signal variable name: reading

Figure 39 - The Custom DMM Configuration Dialog

This dialog allows you to enter the names of three macros that you must supply yourself. The startup macro is called before the sequence, and the shutdown macro is called at the end of the sequence. The reading macro is called each time a Y reading is required. It should assign the reading value to the variable named in the "Signal variable name" field. The signal name and units should also be entered so that it can be correctly displayed on graphs and stored in data files.

Note that these macros need not just make a simple reading. They can be used to perform any function you require at a sequence of temperature and field points.

5.3 Editing the Sequence

To access the sequence editor, click on the "Edit sequence" button in the Macro Engineer dialog. A list similar to that shown in figure 5.3 will be displayed.

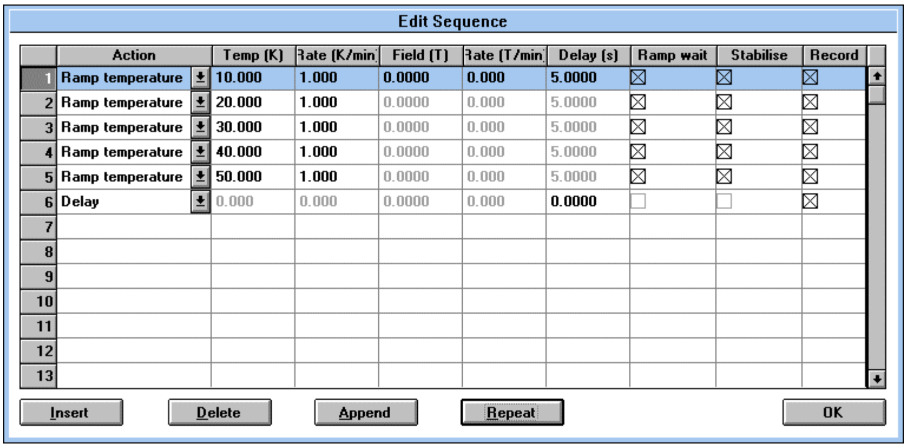
Each row in the table corresponds to a step in the sequence. You can edit the list by using the Insert, Delete and Append buttons. The first column in each row can be used to choose the action performed. You can choose from Delay, Ramp Temperature and Ramp Field. The remaining columns specify the action further.

If you chose Ramp Temperature, you can specify the target temperature and the ramp rate. Similarly, if you chose Field, you can specify the target field and ramp rate.

A delay may be specified after each action. If you want the macro wait for the temperature or field ramp to complete before proceeding, put a click in the "Ramp wait" column. The only time a macro should not wait for a ramp to complete is when you want a temperature and field ramp to execute at the same time.

If you chose a temperature ramp, you can choose whether temperature stability is to be achieved using the temperature stability criteria in section 5.1. Place a cross in the "Stabilise" column to wait for temperature stability before continuing the macro.

Finally, you can choose whether or not a data point is measured for the graph or data file by putting a cross in the column labelled "Record".



	Action	Temp (K)	Rate (K/min	Field (T)	Rate (T/min	Delay (s)	Ramp wait	Stabilise	Record
1	Ramp temperature	10.000	1.000	0.0000	0.000	5.0000	☒	☒	☒
2	Ramp temperature	20.000	1.000	0.0000	0.000	5.0000	☒	☒	☒
3	Ramp temperature	30.000	1.000	0.0000	0.000	5.0000	☒	☒	☒
4	Ramp temperature	40.000	1.000	0.0000	0.000	5.0000	☒	☒	☒
5	Ramp temperature	50.000	1.000	0.0000	0.000	5.0000	☒	☒	☒
6	Delay	0.000	0.000	0.0000	0.000	0.0000	☐	☐	☒
7									
8									
9									
10									
11									
12									
13									

Figure 40 - Editing a Sequence

The button labelled "repeat" is used to generate a number of entries in the table for performing the same operation over a range of values. For example, the first five rows in Figure 40 were generate automatically using this feature.

6 ObjectBench Instrument Drivers

This chapter contains a section describing each instrument driver in your system. Instrument drivers are supplied in separate files, with extension ".drv", contained within the main ObjectBench program. There are two kinds of instrument drivers, real and virtual. Real instrument drivers are most common and correspond to a physical instrument, such as a temperature controller or power supply. Virtual instrument drivers do not control real instruments, but acquire their inputs from real instrument drivers. Typically they combine the inputs from the real instrument drivers to perform calculations which help to interpret the results of an experiment. They are generally only present in complete experiment systems supplied by Oxford Instruments, such as the Faraday Balance or VSM systems.

In general, different combinations of drivers may be supplied with different systems. For this reason, the contents of this chapter may vary from system to system.

Section 4.2.1 describes how to load an instrument driver into ObjectBench.

6.1 Interfacing Instruments to ObjectBench

Interfacing to the instrument can be via RS232, ISOBUS, GPIB or GPIB Gateway interface. Note that the GPIB interface is optional and may not be fitted.

The RS232 interface is a standard serial connection that may be used to connect a single instrument to one serial port on the PC using a standard 25 way pin to pin connection. The ISOBUS interface allows more than one instrument to be connected to a single serial port on the PC, using a special lead supplied by Oxford Instruments. The GPIB is a standard parallel interface that allows a number of instruments to be connected to a single GPIB adapter card in the PC using a standard GPIB cable. This card must be the type supplied by CEC, with part number PC488. The GPIB Gateway interface allows a single GPIB instrument to act as an ISOBUS master. The alternative methods of interfacing are illustrated in Figure 41 and described more fully in the instrument's manual. Choose the scheme which is most suited to your needs and connect the PC and instrument accordingly.

If you are using a serial port on your PC it may be necessary to use a 9 to 25 pin converter. See your computer's documentation for further details of its COM ports.

Having physically connected the instrument to the PC, you should now configure the instrument's interface in software. Select the instrument menu item "Config"/"Interface" to display the dialog shown in Figure 42.

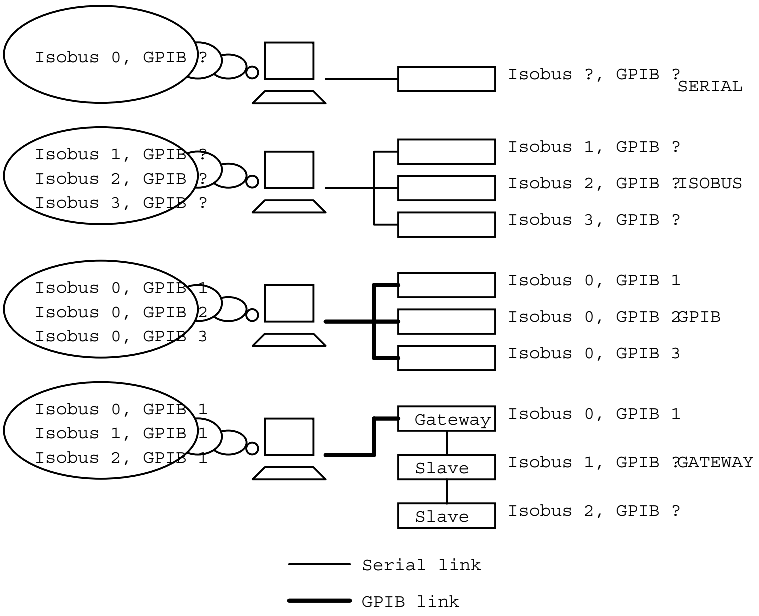
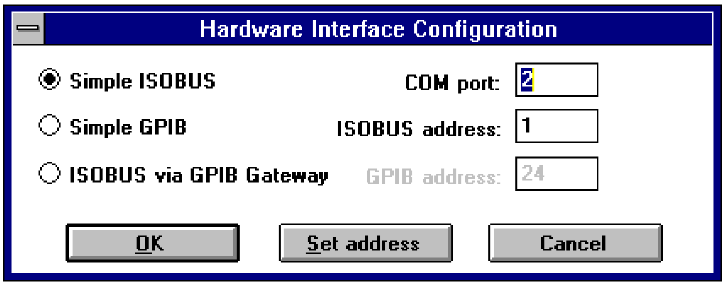


Figure 41 Alternative interfacing methods.

Click next to the interfacing option you require on the left of the dialog: select ISOBUS, GPIB or Gateway. If you require a simple RS232 interface select ISOBUS. Then enter the PC's COM port and the instrument's ISOBUS and GPIB addresses that are to be used in the remaining fields in the dialog. If you have selected ISOBUS or GPIB Gateway you should now set the instrument's ISOBUS address by clicking on the "Set address" button. **It is important that no two instruments on the same ISOBUS or GPIB bus share the same address, or communication will be impossible.**


Hardware Interface Configuration

● Simple ISOBUS
○ Simple GPIB
○ ISOBUS via GPIB Gateway

COM port: 2
ISOBUS address: 1
GPIB address: 24

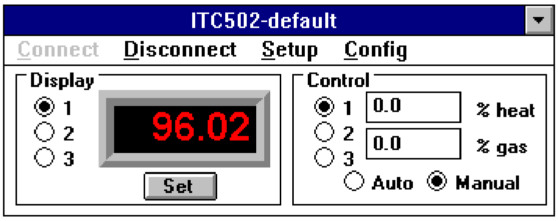
Figure 42 The interface configuration dialog box.

6.2 ITC502 Temperature Controller

The Intelligent Temperature Controller model 502 (ITC502) is a one (optionally three) channel temperature controller and monitor designed specifically for cryogenic system control. Its ObjectBench instrument driver allows full remote control, including control of set point, PID control, and ramping. It also allows custom sensor ranges to be designed and downloaded to the instrument. Full details of the instrument are included in a separate manual. The name of the ObjectBench driver is "itc502.drv".

6.2.1 Preparing to Use the ITC502 from ObjectBench

Figure 43 shows the ITC502 instrument driver main window which is displayed by double clicking on the ITC502 icon.



Display: ● 1 ○ 2 ○ 3 — 96.02

Control: ● 1 ○ 2 ○ 3

0.0 % heat

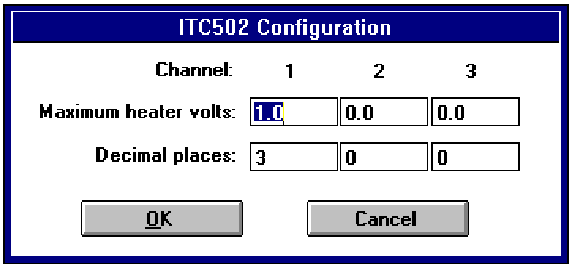
0.0 % gas

○ Auto ● Manual

Figure 43 The ITC502 software front panel.

Before the software front panel can acquire data from the hardware instrument, you must connect (interface) the instrument to the PC and configure the instrument driver correctly as described in section 6.1.

Select the instrument menu item "Config"/"Instrument" to the display the dialog box shown in Figure 44.



Channel:	1	2	3
Maximum heater volts:	1.0	0.0	0.0
Decimal places:	3	0	0

Figure 44 The ITC502 configuration dialog.

The fields in the ITC502 Instrument Configuration dialog allow you to specify the maximum heater voltage and the decimal point position for each channel. The maximum heater volts is the maximum voltage that may be applied to the heater when controlling using a particular channel, and should be set to the maximum safe value permitted for your system. **Note that setting the maximum heater volts too high might damage your system.** If in doubt, contact Oxford Instruments.

The instrument does not return the position of the decimal point in the data, so it is necessary to define it for each channel. The position of the decimal point for each channel may be determined by displaying each channel in turn on the front panel of the ITC502. Enter the values in the dialog so that the software may correctly interpret data from the instrument.

Finally, you may "connect" to the ITC502 by clicking on the ITC502's "Connect" menu option. The current temperature should appear on the display and be updated in real time. If this is not the case, an error message will be displayed which should give an indication of the nature of problem. The most likely cause of difficulty is in the interfacing options - if in doubt, review these first.

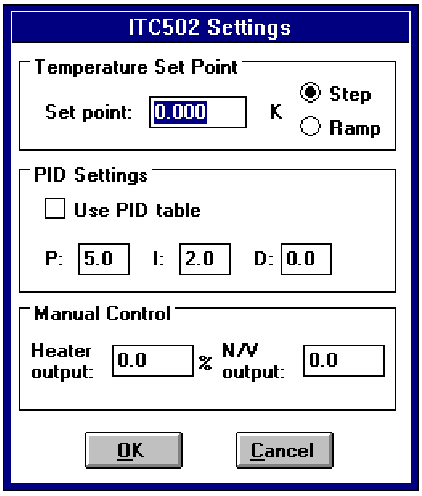
6.2.2 Operating the ITC502 from ObjectBench

The ITC502 software control panel is shown in Figure 43. It is very similar to the physical instrument front panel and operates in a similar way.

The temperature channel to display is chosen by clicking on the channel 1, 2 or 3 radio button in the "Display" box. The control channel is selected by clicking on the radio button labelled 1, 2 or 3 in the "Heater" box. Automatic or manual temperature control is selected by clicking on the Auto or Manual radio buttons in the "Heater" box. The current heater output is always displayed in percent. For a single channel instrument the data for channels 2 and 3 is not meaningful.

While one of the ITC502's front panel buttons is pressed by a user, the ITC502 delays responding to data requests by the PC. ObjectBench warns you when this situation occurs by sounding a two tone warning alarm through the PC's loudspeaker. This warning is a cue to the user to remove his/her finger! If the button continues being pressed, ObjectBench will time out after 5 seconds.

To change the set point, the manual heater output or the PID settings, click on the Set button to reveal the dialog shown in Figure 45.



ITC502 Settings

Temperature Set Point

Set point: 0.000 K

● Step

○ Ramp

PID Settings

□ Use PID table

P: 5.0 I: 2.0 D: 0.0

Manual Control

Heater output: 0.0 %

N/V output: 0.0

Figure 45 The ITC502 settings dialog.

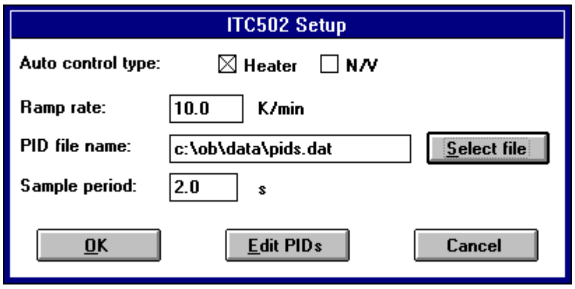
The temperature set point may be entered in the first field of the dialog. When the ITC502 is in manual control mode, the heater and needle valve outputs may be entered in %. The "Use PID table" box should be checked if you wish to use a PID vs. temperature table as described below. Otherwise, the P, I and D settings may be entered in their respective fields.

The set point may be stepped or ramped from the existing set point at a constant ramp rate. Select one of these options by clicking on the "Step" or "Ramp" radio button. The constant ramp rate is chosen in the general setup dialog described below.

When you have finished using the ITC502 from ObjectBench, click on the Disconnect main menu option to stop communication with the instrument.

6.2.3 The ITC502 General Setup Dialog

The General Setup dialog is accessed by selecting the instrument's "Setup"/"General Setup" menu option and displays the dialog shown in Figure 46.



ITC502 Setup

Auto control type: ☒ Heater ☐ N/V

Ramp rate: 10.0 K/min

PID file name: c:\ob\data\pids.dat

Sample period: 2.0 s

Figure 46 The ITC502 general setup dialog.

This dialog allows you to specify the control type, that is heater, needle valve, or both. Check the appropriate boxes in the dialog - note that you must check at least one box.

The sampling period is the rate at which new readings are obtained from the instrument and the display updated. A typical value is 2 seconds.

It is possible for the ITC502 to use a table of PID values and temperature ranges supplied by the user, so that whenever the temperature set point is changed, suitable PID values from the table are set up. This information is contained in an ObjectBench format data file which is named in the "PID file name" field. The file specification should include the full path, as shown. To select an existing PID file, click on the "Select File" button and choose the file from the list presented. If you wish to create a new PID file, just type its name in this field and edit it. To edit the PID file, click on the "Edit PIDs" button.

The ramp rate used in ramp mode is entered in kelvin per minute in the appropriate field in this dialog. Ramp mode is selected in the settings dialog described above.

6.2.4 The ITC502 RAM Setup Dialog

The instrument RAM setup dialog is shown in Figure 47. This dialog allows the complete instrument RAM to be read and written, and also loaded and saved to disk files. This feature allows one or more complete ITC502 configurations to be saved and restored. In addition, it is possible to manipulate the instrument's custom linearisations while a RAM map is loaded in the dialog and assign a range to a channel. The example in Figure 47 shows one custom range in use, with the symbol "rF.52". Custom ranges can also be saved and loaded into the dialog's current RAM.

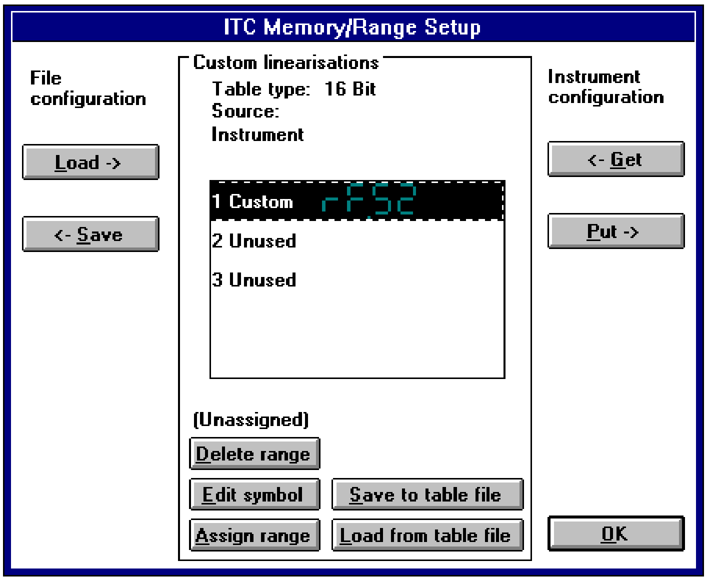


Table type: 16 Bit
Source:
Instrument

1 Custom
2 Unused
3 Unused

(Unassigned)

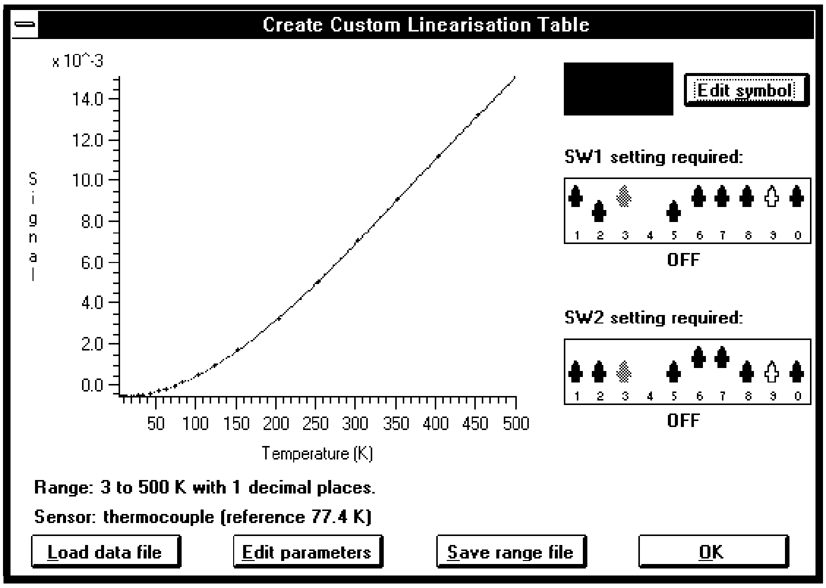
Figure 47 The ITC502 RAM configuration dialog.

To read the instrument's RAM contents into the computer, click on the "<- Get" button on the right of the dialog. Alternatively, this data may be read from a previously stored disk file by clicking on the "Load ->" button on the left of the dialog. In either case, when RAM has been loaded into the dialog, the central box in the dialog will contain information about its custom linearisations. The table type (16 or 24 bits) and the RAM map source (instrument or file) are also displayed. To transfer the RAM map back to the instrument or to a disk file, click on the "Put ->" or "<- Save" buttons respectively. In this way, RAM maps may be transferred between disk files and the instrument in a flexible way.

When a RAM map is present in the dialog, it is possible to edit its custom linearisation tables in a number of ways. To operate on a particular table, select it using the mouse - at this point, available buttons will be enabled. Ranges may be written to or read from disk files by selecting the required table with the mouse and clicking on "Save table file" or "Load table file" respectively. It is possible to load range files that have been designed using the "Setup"/"Create Range" menu option described below. You may edit the symbol of the selected range by pressing the "Edit symbol" button, and you may select which sensor channels use the selected range by pressing the "Assign range" button.

6.2.5 The ITC502 Range Design Dialog

The ITC502 range design dialog is accessed by selecting the instrument menu option "Setup"/"Create Range", displaying the dialog box shown in Figure 48.


Create Custom Linearisation Table

SW1 setting required:

SW2 setting required:

Range: 3 to 500 K with 1 decimal places.

Sensor: thermocouple (reference 77.4 K)

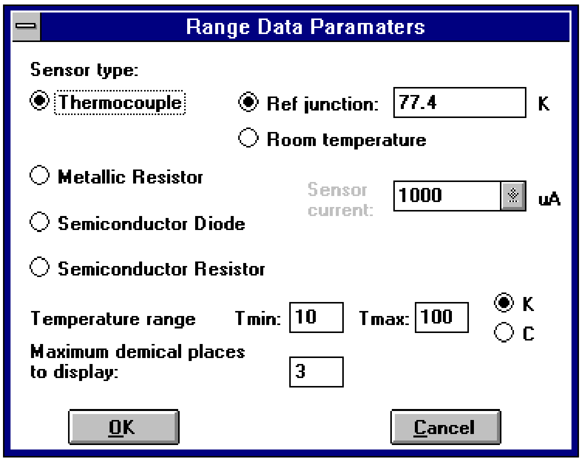
Figure 48 The ITC502 custom range design dialog.

This dialog may be used to convert raw sensor calibration data into the format required by the instrument ready to be downloaded into the instrument RAM setup dialog (see above). The data should be presented as a standard ObjectBench format data file (see section 9) with a column for temperature data and a column for sensor data (resistance, voltage etc.). A number of sample data files are provided in a subdirectory named \ob\data\sensors.

First, you must load the sensor data file using the "Load data file" button. If the file is in the accepted format, a further dialog will be displayed asking for further information about the sensor to be used. This dialog is shown in Figure 49.

Select the sensor type by clicking on the radio button labelled "Thermocouple", "Metallic Resistor", "Semiconductor Diode" or "Semiconductor Resistor" (See the ITC502 manual for further information on sensor types). Then, enter further data about the sensor in the remaining fields which are not disabled. For example, Figure 49 shows that a thermocouple sensor is to be used with a reference junction at 77.4K.

The fields at the bottom of the dialog allow the temperature range and decimal places displayed to be selected. When you are happy with your selection, click on "OK". You may return to this dialog at any time to change the settings by clicking on the "Edit parameters" button.



Range Data Paramaters

Sensor type:

● Thermocouple

● Ref junction: 77.4 K

○ Room temperature

○ Metallic Resistor

Sensor current: 1000 uA

○ Semiconductor Diode

○ Semiconductor Resistor

Temperature range Tmin: 10 Tmax: 100 ● K ○ C

Maximum demical places to display: 3

Figure 49 The sensor data dialog box.

At this point, a graph of a function of your raw data (shown as red crosses) will be displayed against temperature. The function of the data used depends on the sensor type, and is intended to make the curve more nearly linear with a positive gradient. Also on this graph will appear a fit to the raw data, shown as blue dots. This is the data that will actually be downloaded to the instrument. To save the custom range data, click on the "Save Range File" button.

When a custom range has been created, the input card DIP switches in the instrument should be set up to correspond to the settings shown on the right of the dialog.

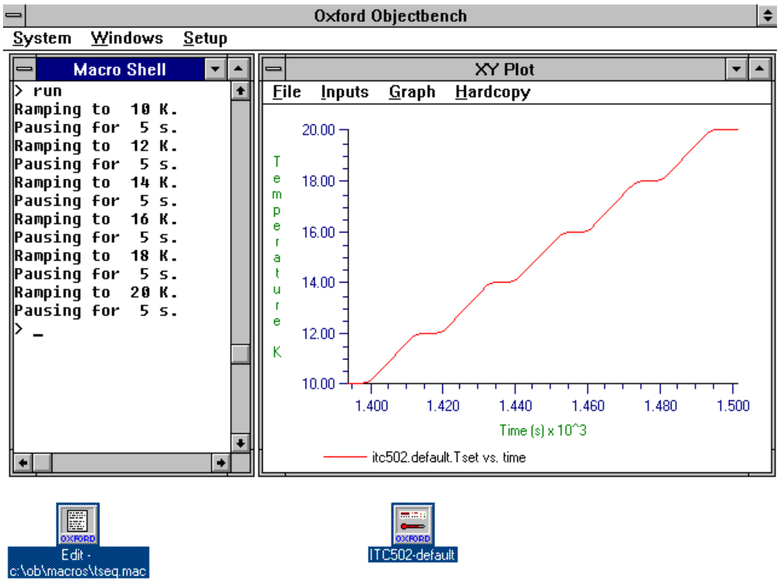
You may edit the symbol for the range by pressing the "Edit symbol" button.

Note that it may not always be possible to find suitable switch settings for the sensor and range that you have requested. This occurs when the sensor input gain or offset required in electronics exceeds that possible. It is then necessary to choose a different temperature range.

6.2.6 Automatic Temperature Sequencing

The ITC502 contains commands which can be used from the macro language to set up a sequence of temperature ramps, steps, and delays. Specifically, the commands are ITC502.ramp and ITC502.step, which select ramp or step set point mode. There are also two instruments variables. ITC502.ramprate and ITC502.isramping. The former allows a ramp rate in kelvin per minute to be set up, and the latter can be monitored to see whether the instrument is currently ramping.

An example macro which sets up a series of temperatures is shown in Figure 51, and a corresponding output screen shown in Figure 50.



```
> run
Ramping to  10 K.
Pausing for  5 s.
Ramping to  12 K.
Pausing for  5 s.
Ramping to  14 K.
Pausing for  5 s.
Ramping to  16 K.
Pausing for  5 s.
Ramping to  18 K.
Pausing for  5 s.
Ramping to  20 K.
Pausing for  5 s.
> _
```

Figure 50 Sequence of temperature set points generated by macro TSEQ.

6.2.7 ITC502 Instrument Signals and Commands

Refer to section 7.10 for details of the signals and commands.

6.3 ITC503 Temperature Controller

The Intelligent Temperature Controller model 503 (ITC503) is a one (optionally three) channel temperature controller and monitor designed specifically for cryogenic system control. Its ObjectBench instrument driver allows full remote control, including control of set point, PID control, and ramping. It also allows custom sensor ranges to be designed and downloaded to the instrument. One feature unique to the ITC503 driver is PID autotuning, which permits automatic determination of the best PID settings for controlling your system. Full details of the instrument are included in a separate manual. The name of the ObjectBench driver is "itc503.drv".

Tip Do not confuse "Auto PID" with "Auto Tuning". The ITC503 front panel has a button labelled "Auto PID" which makes the instrument set P, I and D automatically from an internal table of values. "Auto Tuning" is the process by which this software determines the best values of P, I and D to use under given circumstances. These values may optionally be downloaded to the instrument in the form of a table for use with "Auto PID".

```
          tstart = 10          ; rem ramp from 10 K to
          tfinish = 20         ; rem 100K in steps of
          tstep = 2  ; rem 10 K.
          dwell = 5 ; rem Stop for 5s at each temperature.
          ramprate = 60        ; rem Ramp at 10 K/min.
rem Initialise the instrument:
          itc502.connect
          itc502.auto
rem step to the first temperature, 10K:
          itc502.step
          itc502.tset = tstart
rem ramp to a series of temperatures up to 100K
          itc502.ramp
          ttarget = tstart
loop:
          gosub ramp
          gosub delay
          ttarget = ttarget + tstep
          if ttarget <= tfinish then goto loop
          end
rem A subroutine that delays "dwell" seconds:
delay:    t1 = time
          print "Pausing for ", dwell, " s."
delay1:   if time - t1 < dwell * 1000 then goto delay1
          return
rem A subroutine that ramps from the current set point to "ttarget":
ramp:     itc502.tset = ttarget
          print "Ramping to ", ttarget, " K."
ramp1:    if itc502.isramping then goto ramp1
          return
```

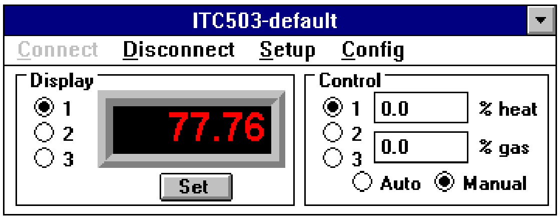
Figure 51 TSEQ, an example macro which generates a sequence of temperature set point ramps and delays for ITC 502

6.3.1 Preparing to Use the ITC503 from ObjectBench

Figure 50 shows the ITC503 instrument driver main window which is displayed by double clicking on the ITC503 icon.

Before the software front panel can acquire data from the hardware instrument, you must connect (interface) the instrument to the PC and configure the instrument driver correctly as described in section 6.1.

Select the instrument menu item "Config"/"Instrument" to the display the dialog box shown in Figure 53.

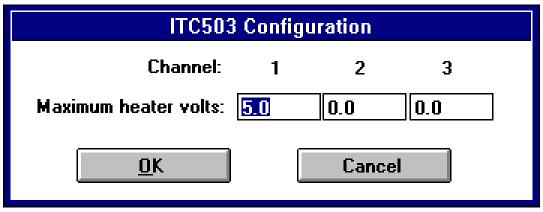


Display
● 1
○ 2
○ 3
77.76

Control
● 1 0.0 % heat
○ 2
○ 3 0.0 % gas
○ Auto ● Manual

Figure 52 The ITC503 software front panel.

The fields in the ITC503 Instrument Configuration dialog allow you to specify the maximum heater voltage for each channel. The maximum heater volts is the maximum voltage that may be applied to the heater when controlling using a particular channel, and should be set to the maximum safe value permitted for your system. **Note that setting the maximum heater volts too high can damage your system.** If in doubt, contact Oxford Instruments.



Channel:	1	2	3
Maximum heater volts:	5.0	0.0	0.0

Figure 53 The ITC503 configuration dialog.

Finally, you may "connect" to the ITC503 by clicking on the ITC503's "Connect" menu option. The current temperature should appear on the display and be updated in real time. If this is not the case, an error message will be displayed which should give an indication of the nature of problem. The most likely cause of difficulty is in the interfacing options - if in doubt, review these first.

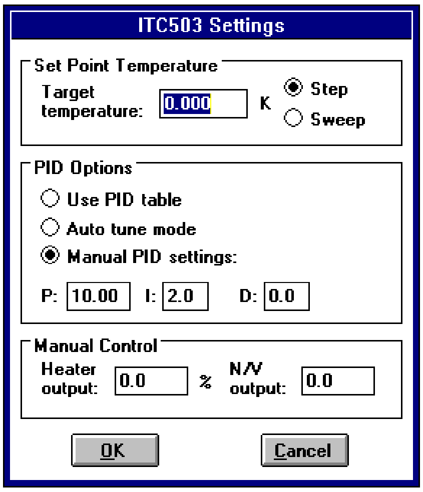
6.3.2 Operating the ITC503 from ObjectBench

The ITC503 software control panel is shown in Figure 52. It is very similar to the physical instrument front panel and operates in a similar way.

The temperature channel to display is chosen by clicking on the channel 1, 2 or 3 radio button in the "Display" box. The control channel is selected by clicking on the radio button labelled 1, 2 or 3 in the "Heater" box. Automatic or manual temperature control is selected by clicking on the Auto or Manual radio buttons in the "Heater" box. The current heater output is always displayed in percent.

While one of the ITC503's front panel buttons is pressed by a user, the ITC503 delays responding to data requests by the PC. ObjectBench warns you when this situation occurs by sounding a two tone warning alarm through the PC's loudspeaker. This warning is a cue to the user to remove his/her finger! If the button continues to be pressed, ObjectBench will time out after 5 seconds.

To change the set point, the manual heater output or the PID settings, click on the Set button to reveal the dialog shown in Figure 54.



ITC503 Settings

Set Point Temperature

Target temperature: 0.000 K

● Step

○ Sweep

PID Options

○ Use PID table

○ Auto tune mode

● Manual PID settings:

P: 10.00 I: 2.0 D: 0.0

Manual Control

Heater output: 0.0 %

N/V output: 0.0

Figure 54 The ITC503 Settings Dialog.

The temperature set point may be entered in the first field of the dialog. When the ITC503 is in manual control mode, the heater and needle valve outputs may be entered in %.

There are three options for the PID settings, "Use PID Table", "Auto Tune Mode", and "Manual PID Settings".

The "Manual PID Settings" option allows P, I and D settings to be entered manually in their respective fields (see the ITC503 manual for information on how to choose values for P, I and D manually). The new values will take effect at once.

The "Use PID Table" option should be checked if you wish to use a PID vs. temperature table as described below in section 6.3.3. This feature allows P, I and D to be determined from a table of temperature ranges and PID settings, and updated automatically whenever a new temperature set point is entered.

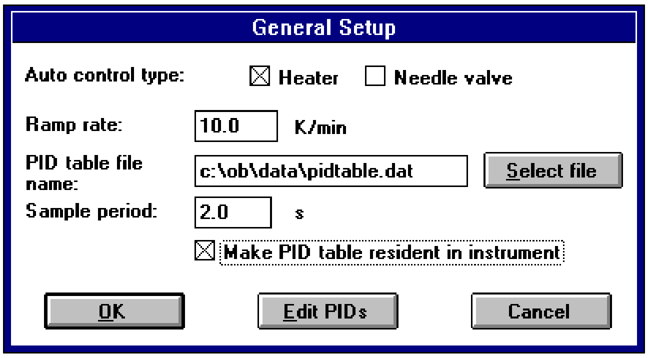
The "Auto Tune Mode" option should be selected if you wish the computer to determine the best possible values for P, I and D automatically. See section 6.3.6 below for further details.

The set point may be stepped or ramped from the existing set point at a constant ramp rate. Select one of these options by clicking on the "Step" or "Ramp" radio button. When "Ramp" mode is selected, the temperature field displays the target of the ramp; otherwise, in "Step" mode, it displays the current temperature set point. The constant ramp rate is chosen in the general setup dialog (see below).

When you have finished using the ITC503 from ObjectBench, click on the Disconnect main menu option to stop communication with the instrument.

6.3.3 The ITC503 General Setup Dialog

The General Setup dialog is accessed by selecting the instrument's "Setup"/"General Setup" menu option and displays the dialog shown in Figure 55.



General Setup

Auto control type: ☒ Heater ☐ Needle valve

Ramp rate: 10.0 K/min

PID table file name: c:\ob\data\pidtable.dat

Sample period: 2.0 s

☒ Make PID table resident in instrument

Figure 55 The ITC503 general setup dialog.

This dialog allows you to specify the control type, that is heater, needle valve, or both. Check the appropriate boxes in the dialog - note that you must check at least one box.

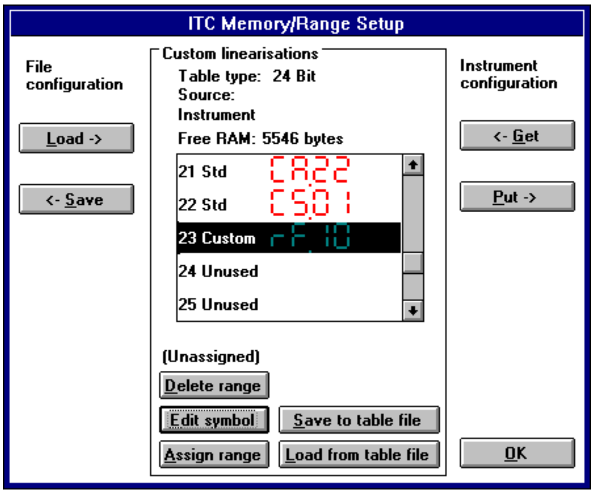
The sampling period is the rate at which new readings are obtained from the instrument and the display updated. A typical value is 2 seconds.

It is possible for the ITC503 to use a table of PID values and temperature ranges supplied by the user, so that whenever the temperature set point is changed, suitable PID values from the table are set up. This information is contained in an ObjectBench format data file which is named in the "PID file name" field. The file specification should include the full path, as shown. To select an existing PID file, click on the "Select File" button and choose the file from the list presented. If you wish to create a new PID file, just type its name in this field and edit it. To edit the PID file, click on the "Edit PIDs" button. The PID table selected will not actually be used until the "Use PID Table" option is selected in the ITC503 Settings dialog (see section 6.3.2).

If the "Make PID table resident in instrument" option is selected, the PID table will be downloaded to the ITC503 from the computer and its "Auto PID" mode used to select P, I and D settings for a given set point temperature. Otherwise, the PID table is held in the computer which updates P, I and D whenever a new set point is entered through the computer. You may wish to make the PID table resident in the instrument if you wish to use it separately from the computer.

The ramp rate used in ramp mode is entered in kelvin per minute in the appropriate field in this dialog. Ramp mode is selected in the settings dialog described above.

6.3.4 The ITC503 RAM Setup Dialog


ITC Memory/Range Setup

Custom linearisations
Table type: 24 Bit
Source:
Instrument
Free RAM: 5546 bytes

21 Std CA.22
22 Std CS.01
23 Custom rF.10
24 Unused
25 Unused

(Unassigned)

Figure 56 The ITC503 RAM configuration dialog.

The instrument RAM setup dialog is shown in Figure 56. This dialog allows the complete instrument RAM to be read and written, and also loaded and saved to disk files. This feature allows one or more complete ITC503 configurations to be saved and restored. In addition, it is possible to manipulate the instrument's custom linearisations while a RAM map is loaded in the dialog and assign a range to a channel. The example in Figure 56 shows some standard ranges, one custom range and some unused ranges. The list may be scrolled to see more ranges: it is possible to have up to 32 ranges in total in the ITC503. Custom ranges can also be saved and loaded into the dialog's current RAM.

To read the instrument's RAM, click on the "<- Get" button on the right of the dialog. To read from a RAM file, click on the "Load ->" button on the left of the dialog. In either case, when RAM has been loaded into the dialog, the central box in the dialog will contain information about its custom linearisations. The table type (16 or 24 bits) and the RAM map source (instrument or file) are also displayed. To transfer the RAM map to the instrument or to a disk file, click on the "Put ->" or "<- Save" buttons respectively. In this way, RAM maps may be transferred between disk files and the instrument in a flexible way.

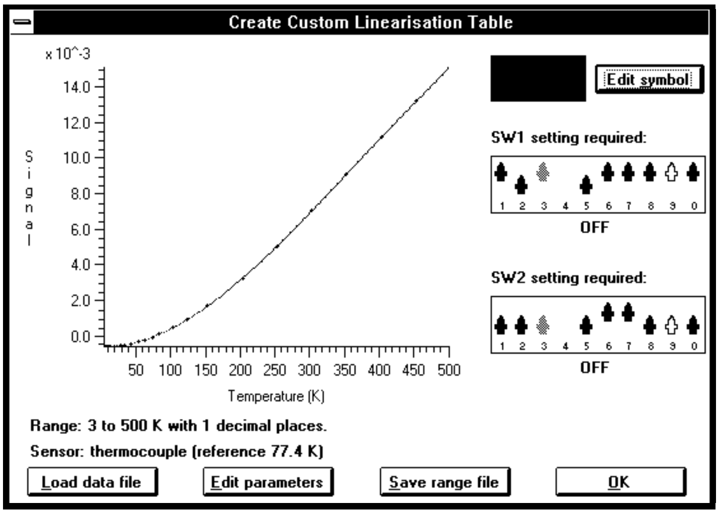
When a RAM map is present in the dialog, it is possible to edit its custom linearisation tables in a number of ways. To operate on a particular table, select it using the mouse - at this point, available buttons will be enabled. Ranges may be written to or read from disk files by selecting the required table with the mouse and clicking on "Save table file" or "Load table file" respectively. You may load a new range into either a "custom" or an "unused" entry - the standard ("Std") ranges may not be overwritten with new ranges. At the top of the dialog box, the free RAM in the instrument is shown. This figure decreases as custom ranges are loaded into the instrument, until it is not possible to load a new range. Typically, there is room for about 8 custom ranges, but this depends on the size of the range data used.

Range files may be designed using the "Setup"/"Create Range" menu option described in section 6.3.5. You may edit the symbol of the selected range by pressing the "Edit symbol" button, and you may select which sensor channels use the selected range by pressing the "Assign range" button.

When you have downloaded new RAM contents to the instrument, the software will prompt you to transfer the RAM contents to EEPROM for permanent storage when the instrument is switched off.

6.3.5 The ITC503 Range Design Dialog

The ITC503 range design dialog is accessed by selecting the instrument menu option "Setup"/"Create Range", displaying the dialog box shown in Figure 57.



SW1 setting required:

OFF

SW2 setting required:

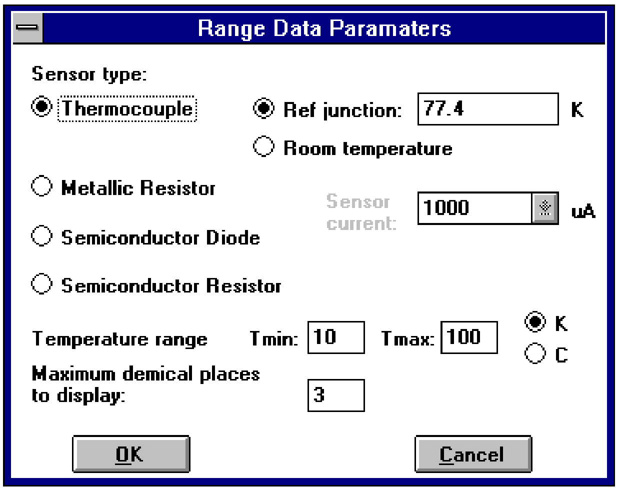
OFF

Range: 3 to 500 K with 1 decimal places.

Sensor: thermocouple (reference 77.4 K)

Figure 57 The ITC503 custom range design dialog.

This dialog may be used to convert raw sensor calibration data into the format required by the instrument ready to be downloaded into the instrument RAM setup dialog (see above). The data should be presented as a standard ObjectBench format data file (see section 9) with a column for temperature data and a column for sensor data (resistance, voltage etc.). All the data files for the standard ranges are provided in a subdirectory named \ob\data\sensors as samples.



Sensor type:

● Thermocouple ● Ref junction: 77.4 K

○ Room temperature

○ Metallic Resistor Sensor current: 1000 uA

○ Semiconductor Diode

○ Semiconductor Resistor

Temperature range Tmin: 10 Tmax: 100 ● K ○ C

Maximum demical places to display: 3

Figure 58 The sensor data dialog box.

First, you must load the sensor data file using the "Load data file" button. If the file is in the accepted format, a further dialog will be displayed asking for further information about the sensor to be used. This dialog is shown in Figure 58.

Select the sensor type by clicking on the radio button labelled "Thermocouple", "Metallic Resistor", "Semiconductor Diode" or "Semiconductor Resistor" (See the ITC503 manual for further information on sensor types). Then, enter further data about the sensor in the remaining fields which are not disabled. For example, Figure 58 shows that a thermocouple sensor is to be used with a reference junction at 77.4K.

The fields at the bottom of the dialog allow the temperature range and maximum number of decimal places displayed to be selected. When you are happy with your selection, click on "OK". You may return to this dialog at any time to change the settings by clicking on the "Edit parameters" button.

At this point, a graph of a function of your raw data (shown as red crosses) will be displayed against temperature. The function of the data used depends on the sensor type, and is intended to make the curve more nearly linear with a positive gradient. Also on this graph will appear a fit to the raw data, shown as a blue line. This is the data that will actually be downloaded to the instrument. To save the custom range data, click on the "Save Range File" button.

When a custom range has been created, the input card DIP switches in the instrument should be set up to correspond to the settings shown on the right of the dialog.

You may edit the symbol for the range by pressing the "Edit symbol" button.

Note that it may not always be possible to find suitable switch settings for the sensor and range that you have requested. This occurs when the sensor input gain or offset required in electronics exceeds that possible. It is then necessary to choose a different temperature range.

6.3.6 Auto Tuning PIDs

Auto tuning PIDs is the process of finding the best P, I and D settings to use to control a particular system at a particular temperature. A manual method for finding P, I and D is described in the ITC503 manual.

The best values of P, I and D to use depend on a number of factors. The cryogenic system attenuation and time constant, which depend on temperature, determine the usable ranges of values for P and I respectively. Another important factor is the maximum heater voltage in use, which affects P. Finally, the shape of the required response determines P. For example, if a fast response with some overshoot is required, P will be lower than when no overshoot is permitted. Thus, a PID setting is applicable to a given system at a given temperature, using a given maximum heater voltage and a particular overshoot criterion. If a system is intended to be used over a wide temperature range, more than one PID setting will need to be used, dependent on temperature.

To auto tune a system, the ITC503 driver must be put into "Auto tune PID" mode and then the system exercised by cycling its set point. After each change in set point, the driver automatically refines the PID settings in use, so that after a number of cycles, the best settings have been found. This process can be performed automatically by use of a custom application and macros supplied with ObjectBench.

To autotune your system, first use the main ObjectBench menu option "Windows"/"Applications" to load the custom application called "autotune.app". You should see the dialog box shown in Figure 59.

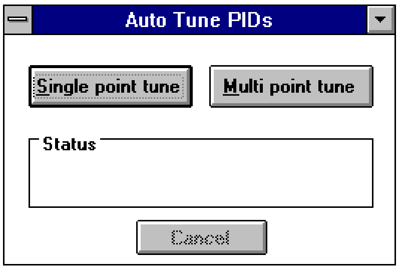
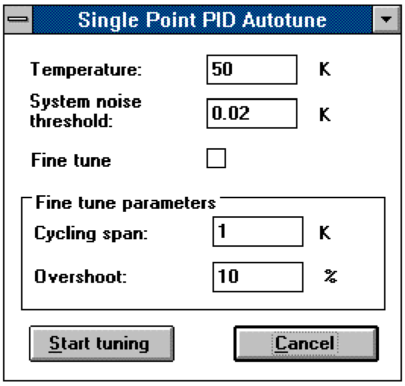


Figure 59 The Auto Tuning Dialog Box.

To perform PID tuning at a single point, click on the "Single point tune" button. A new dialog, shown in Figure 60, will be displayed.

Before autotuning the system, you should choose the gas flow rate (if applicable) and the maximum heater output voltage that you wish to work with. The maximum heater voltage should be chosen so that the heating power with maximum heater volts is similar to the maximum cooling power in the temperature range of interest. Note that setting the maximum heater volts too high can damage the heater in some systems. If in doubt contact Oxford Instruments.



Temperature: 50 K

System noise threshold: 0.02 K

Fine tune □

Fine tune parameters

Cycling span: 1 K

Overshoot: 10 %

Figure 60 The single point auto tune dialog.

You should also assess the basic noise level of the temperature measurement by observing the readout on the instrument. Then, choose a step size for the autotune macro. This is the size of the temperature step that the macro will use for exercising the system. For best results, the step size should be chosen to be at least 10 times bigger than the noise level, but not so big that the heater output saturates (that is reaches 0% or 100%) during the step. If the heater output saturates, the rising and falling transients will not be symmetrical and auto tune will converge on compromise settings which are not optimum.

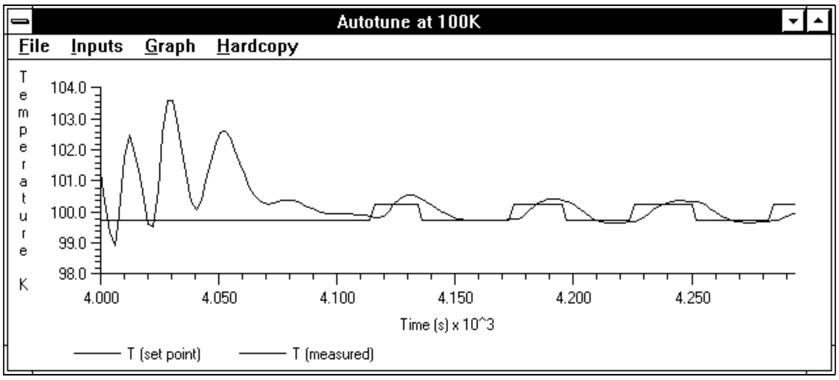


Figure 61 An example of a fine auto tuning sequence at 100K. Notice how the overshoot is converging on 10%, the value required.

Enter the temperature that you wish to auto tune at (50K, in this example), and the cycling span chosen. The system noise threshold is the minimum temperature change that will be noticed by the auto tune algorithm, and should be chosen to be just greater than the basic noise level assessed previously. Finally, enter the overshoot that is to be permitted, as a percentage of the temperature step. If a larger overshoot is permitted, the response to set point changes will be fast, while a small overshoot results in a slow system response. In general, a value of about 10% has been found to be a good compromise. Press "Start tuning" to start the auto tuning algorithm. When the algorithm has finished, the instrument PIDs will have been set to their optimum values. A typical auto tuning sequence is shown in Figure 61. This might take between 2 minutes and an hour depending on the system time constant.

You can choose whether to perform fine tuning or not. If fine tuning is selected autotuning will take longer but will result in values for P, I and D which result in an overshoot closer to that requested.

To perform multi point tuning over a range of temperatures, you should click on the "Multi point tune" button shown in Figure 59. This will display the dialog box shown in Figure 62. This dialog is similar to the single point auto tune dialog above, but some features deserve further explanation.

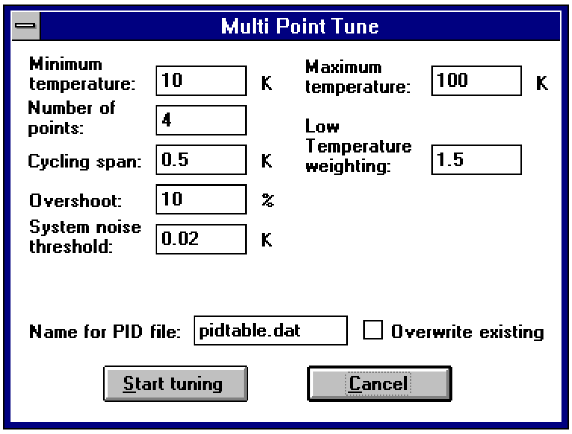
Instead of entering a single temperature, a minimum and maximum temperature for auto tuning has to be entered. The number of PID settings required in the range is entered in the "Number of points" field.

In a typical cryogenic system, PIDs vary most rapidly at low temperatures, so it is best to autotune at closer temperature intervals in this range. This may be specified using the "Low temperature weighting" field. The value in this field is actually the ratio of successive temperature spans between PID tunings as temperature increases. For example, if the weighting were set to 2, PID spans might be taken at 2K, 4K, 8K, 16K, 32K and so on. The greater the weighting, the greater the clustering of readings at low temperatures. To obtain equally spaced settings, set the weighting to 1. The PIDs will be tuned at the centre of each span, (that is 3K, 6K, 12K and 24K in the above example). The cycling span used at each temperature is the same.

The PID settings found as a result of each tuning are stored in the data file named in the dialog. This data is then suitable for use as a PID table file (see section 6.3.3).

Note that the PID tuning sequences are just ObjectBench macros and custom application dialogs, so they can be adapted to your special needs. See other sections in this manual for more information.

Under some circumstances, the autotuning may not result in good values of P, I and D. This may be recognised by examining the shape of the response to temperature steps near the end of the autotuning sequence, and comparing it with the requested response. Possible reasons for this failure are that the noise threshold has been set too high or too low, or that the temperature step is too large.



Multi Point Tune

Minimum temperature: 10 K

Maximum temperature: 100 K

Number of points: 4

Cycling span: 0.5 K

Low Temperature weighting: 1.5

Overshoot: 10 %

System noise threshold: 0.02 K

Name for PID file: pidtable.dat

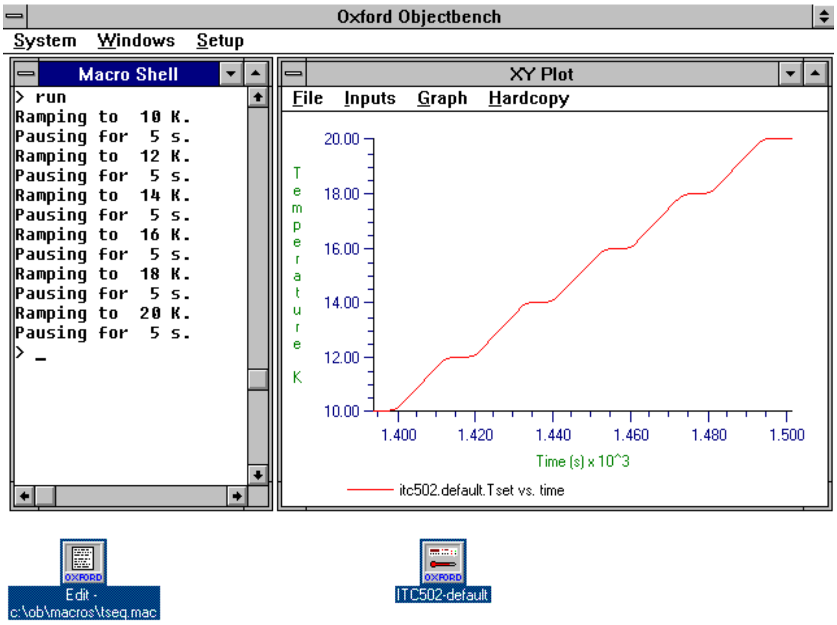
□ Overwrite existing

Figure 62 The multi point auto tuning dialog.

6.3.7 Automatic Temperature Sequencing

The ITC503 contains commands which can be used from the macro language to set up a sequence of temperature ramps, steps, and delays. Specifically, the commands are ITC503.ramp and ITC503.step, which select ramp or step set point mode. There are also two instruments variables. ITC503.ramprate and ITC503.isramping. The former allows a ramp rate in kelvin per minute to be set up, and the latter can be monitored to see whether the instrument is currently ramping.

An example macro which sets up a series of temperatures is shown in Figure 64, and a corresponding output screen shown in Figure 63.



```
> run
Ramping to  10 K.
Pausing for  5 s.
Ramping to  12 K.
Pausing for  5 s.
Ramping to  14 K.
Pausing for  5 s.
Ramping to  16 K.
Pausing for  5 s.
Ramping to  18 K.
Pausing for  5 s.
Ramping to  20 K.
Pausing for  5 s.
> _
```

Figure 63 Sequence of temperature set points generated by macro TSEQ.

6.3.8 ITC503 Instrument Signals and Commands

Refer to section 7.10 for details of the signals and commands.

```
          tstart = 10         ; rem ramp from 10 K to
          tfinish = 20        ; rem 20K in steps of
          tstep = 2           ; rem 2 K.
          dwell = 5           ; rem Stop for 5s at each temperature.
          ramprate = 60       ; rem Ramp at 10 K/min.
rem Initialise the instrument:
          itc503.connect
          itc503.auto
          rem step to the first temperature, 10K:
          itc503.step
          itc503.tset = tstart
rem ramp to a series of temperatures up to 100K
          itc503.ramp
          ttarget = tstart
loop:
          gosub ramp
          gosub delay
          ttarget = ttarget + tstep
          if ttarget <= tfinish then goto loop
          end
rem A subroutine that delays "dwell" seconds:
delay:    t1 = time
          print "Pausing for ", dwell, " s."
delay1:   if time - t1 < dwell * 1000 then goto delay1
          return
rem A subroutine that ramps from the current set point to "ttarget":
ramp:     itc503.tset = ttarget
          print "Ramping to ", ttarget, " K."
ramp1:    if itc503.isramping then goto ramp1
          return
```

Figure 64 TSEQ, an example macro which generates a sequence of temperature set point ramps and delays for ITC503.

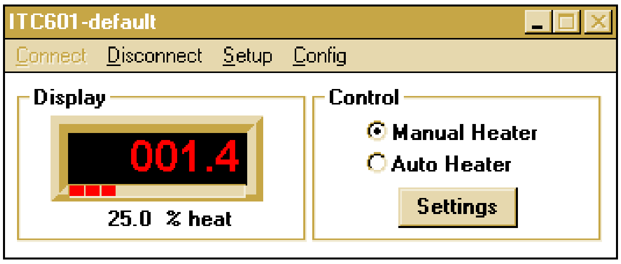
6.4 **ITC**601 **Temperature Controller**

The Intelligent Temperature Controller model 601 (ITC601) is a single channel temperature controller and monitor designed specifically for cryogenic system control. Its ObjectBench instrument driver allows full remote control, including control of set point, PID control, and ramping. It also allows custom sensor ranges to be designed and downloaded to the instrument. One feature unique to the ITC601 instrument is PID autotuning, which permits automatic determination of the best PID settings for controlling your system. Full details of the instrument are included in a separate manual. The name of the ObjectBench driver is "itc601.drv".

Tip Do not confuse "Auto PID" with "Tune". The ITC601 front panel has a button labelled "Auto PID" which makes the instrument set P, I and D automatically from an internal table of values. In contrast, “Tuning” is the process by which the instrument determines the best values of P, I and D to use under given circumstances.

6.4.1 Preparing to Use the ITC601 from ObjectBench

Figure 65 shows the ITC601 instrument driver main window which is displayed by double clicking on the ITC601 icon.



001.4

25.0 % heat

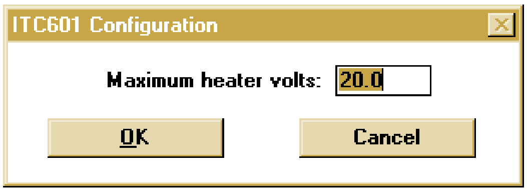
● Manual Heater

○ Auto Heater

Figure 65 The ITC601 software front panel.

Before the software front panel can acquire data from the hardware instrument, you must connect (interface) the instrument to the PC and configure the instrument driver correctly as described in section 6.1.

Select the instrument menu item "Config"/"Instrument" to the display the dialog box shown in Figure 66.



Maximum heater volts: 20.0

Figure 66 The ITC601 configuration dialog.

The maximum heater volts is the maximum voltage that may be applied to the heater when controlling temperature, and should be set to the maximum safe value permitted for your system. **Note that setting the maximum heater volts too high can damage your system.** If in doubt, contact Oxford Instruments.

Finally, you may "connect" to the ITC601 by clicking on the ITC601's "Connect" menu option. The current temperature should appear on the display and be updated in real time. If this is not the case, an error message will be displayed which should give an indication of the nature of problem. The most likely cause of difficulty is in the interfacing options - if in doubt, review these first.

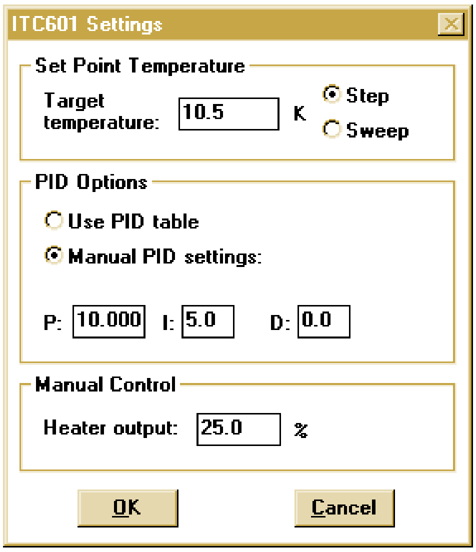
6.4.2 Operating the ITC601 from ObjectBench

The ITC601 software control panel is shown in Figure 65.

The temperature and the heater output are displayed in the left part of the panel. The heater output is displayed as a percentage figure and also as a bar graph below the temperature display. Automatic or manual temperature control is selected by clicking on the “Auto Heater” or “Manual Heater” radio buttons in the "Control" box.

While one of the ITC601's front panel buttons is pressed by a user, the ITC601 delays responding to data requests by the PC. ObjectBench warns you when this situation occurs by sounding a two tone warning alarm through the PC's loudspeaker. This warning is a cue to the user to remove his/her finger! If the button continues to be pressed, ObjectBench will time out after 5 seconds.

To change the set point, the manual heater output or the PID settings, click on the “Settings” button to reveal the dialog shown in Figure 67.



ITC601 Settings

Set Point Temperature

Target temperature: 10.5 K

● Step

○ Sweep

PID Options

○ Use PID table

● Manual PID settings:

P: 10.000 I: 5.0 D: 0.0

Manual Control

Heater output: 25.0 %

Figure 67 The ITC601 Settings Dialog.

The temperature set point may be entered in the first field of the dialog. When the ITC601 is in manual control mode, the heater output may be entered as a percentage of maximum.

There are two options for the PID settings, "Use PID Table" and "Manual PID Settings".

The "Manual PID Settings" option allows P, I and D settings to be entered manually in their respective fields (see the ITC601 manual for information on how to choose values for P, I and D manually). The new values will take effect at once.

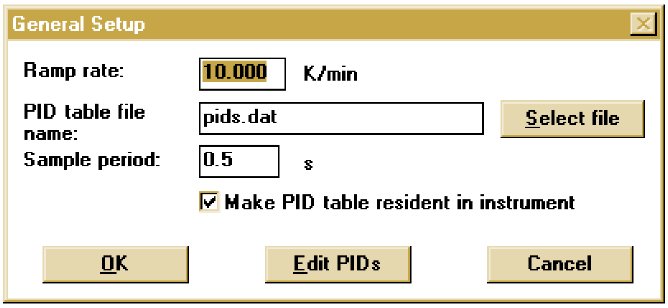
The "Use PID Table" option should be checked if you wish to use a PID vs. temperature table as described below in section 6.4.3. This feature allows P, I and D to be determined from a table of temperature ranges and PID settings, and updated automatically whenever a new temperature set point is entered.

The set point may be stepped or ramped from the existing set point at a constant ramp rate. Select one of these options by clicking on the "Step" or "Ramp" radio button. When "Ramp" mode is selected, the temperature field displays the target of the ramp; otherwise, in "Step" mode, it displays the current temperature set point. The constant ramp rate is chosen in the general setup dialog (see below).

When you have finished using the ITC601 from ObjectBench, click on the Disconnect main menu option to stop communication with the instrument.

6.4.3 The ITC601 General Setup Dialog

The General Setup dialog is accessed by selecting the instrument's "Setup"/"General Setup" menu option and displays the dialog shown in Figure 68.



General Setup

Ramp rate: 10.000 K/min

PID table file name: pids.dat

Sample period: 0.5 s

☑ Make PID table resident in instrument

Figure 68 The ITC601 general setup dialog.

The sampling period is the rate at which new readings are obtained from the instrument and the display updated. A typical value is 2 seconds.

It is possible for the ITC601 to use a table of PID values and temperature ranges supplied by the user, so that whenever the temperature set point is changed, suitable PID values from the table are set up. This information is contained in an ObjectBench format data file which is named in the "PID file name" field. The file specification should include the full path, as shown. To select an existing PID file, click on the "Select File" button and choose the file from the list presented. If you wish to create a new PID file, just type its name in this field and edit it. To edit the PID file, click on the "Edit PIDs" button. The PID table selected will not actually be used until the "Use PID Table" option is selected in the ITC601 Settings dialog (see section 6.4.2).

If the "Make PID table resident in instrument" option is selected, the PID table will be downloaded to the ITC601 from the computer and its "Auto PID" mode used to select P, I and D settings for a given set point temperature. Otherwise, the PID table is held in the computer which updates P, I and D whenever a new set point is entered through the computer. You may wish to make the PID table resident in the instrument if you wish to use it separately from the computer.

The ramp rate used in ramp mode is entered in kelvin per minute in the appropriate field in this dialog. Ramp mode is selected in the settings dialog described above.

6.4.4 The ITC601 Memory Setup Dialog

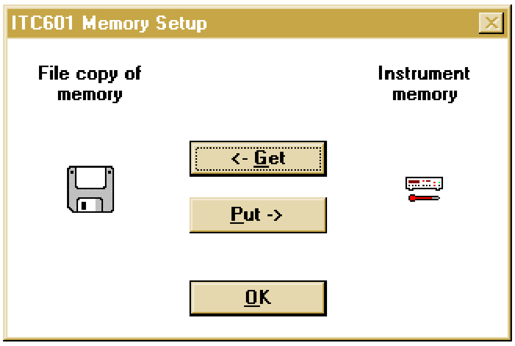


Figure 69 The ITC601 memory setup dialog.

The instrument memory setup dialog is shown in Figure 69. This dialog allows the complete instrument RAM to be transferred to and from disk files. This feature allows one or more complete ITC601 configurations to be saved and restored.

To read the instrument's RAM into a disk file, click on the "<- Get" button. To write from a disk file into the instrument's RAM , click on the " Put ->" button.

When you have downloaded new RAM contents to the instrument, the software will prompt you to transfer the RAM contents to EEPROM for permanent storage when the instrument is switched off.

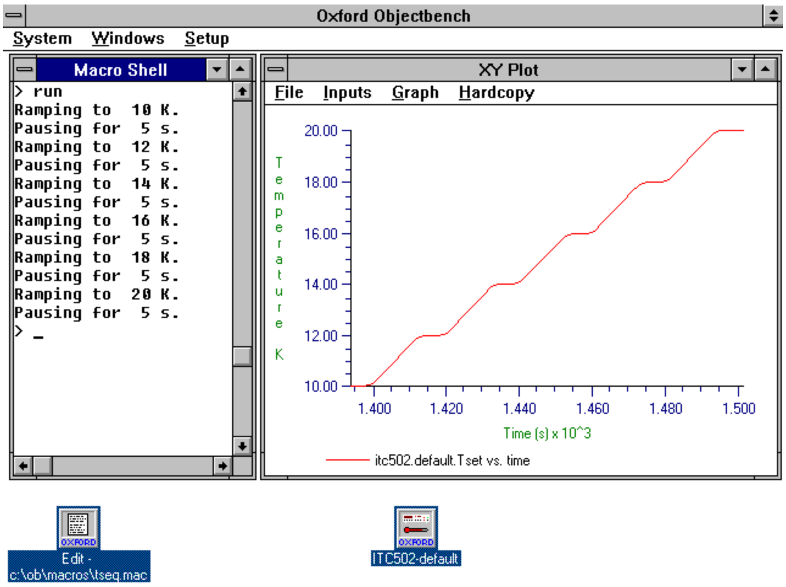
6.4.5 Auto Tuning PIDs

The ITC601 instrument is capable of tuning the values of P, I and D automatically. This process can be started from this software using the menu option “Setup”/”Start PID Tune”. The instrument than enters tuning mode and the instrument controls are greyed out until tuning is complete. To interrupt the tuning process, using the menu option “Setup”/“Cancel PID Tune”.

6.4.6 Automatic Temperature Sequencing

The ITC601 contains commands which can be used from the macro language to set up a sequence of temperature ramps, steps, and delays. Specifically, the commands are ITC601.ramp and ITC601.step, which select ramp or step set point mode. There are also two instruments variables. ITC601.ramprate and ITC601.isramping. The former allows a ramp rate in kelvin per minute to be set up, and the latter can be monitored to see whether the instrument is currently ramping.

An example macro which sets up a series of temperatures is shown in Figure 71, and a corresponding output screen shown in Figure 77.



```
> run
Ramping to  10 K.
Pausing for  5 s.
Ramping to  12 K.
Pausing for  5 s.
Ramping to  14 K.
Pausing for  5 s.
Ramping to  16 K.
Pausing for  5 s.
Ramping to  18 K.
Pausing for  5 s.
Ramping to  20 K.
Pausing for  5 s.
> _
```

Figure 70 Sequence of temperature set points generated by macro TSEQ.

6.4.7 ITC601 Instrument Signals and Commands

Refer to section 7.10 for details of the signals and commands.

6.5 IPS120-10 Magnet Power Supply

The IPS120-10 is a superconducting magnet power supply capable of delivering up 120 amps at up to 10 volts. It is intrinsically a bipolar so that positive and negative fields may be created, and is capable of operating a superconducting switch heater to put the magnet into persistent mode. Its ObjectBench instrument driver allows full remote control, including control of output field or current and ramping rates. Full details of the instrument are included in a separate manual. The name of the ObjectBench driver is "ips120.drv".

```
        tstart = 10         ; rem ramp from 10 K to
        tfinish = 20        ; rem 20K in steps of
        tstep = 2           ; rem 2 K.
        dwell = 5           ; rem Stop for 5s at each temperature.
        ramprate = 60       ; rem Ramp at 10 K/min.
rem Initialise the instrument:
        itc601.connect
        itc601.auto
        rem step to the first temperature, 10K:
        itc601.step
        itc601.tset = tstart
rem ramp to a series of temperatures up to 100K
        itc601.ramp
        ttarget = tstart
loop:
        gosub ramp
        gosub delay
        ttarget = ttarget + tstep
        if ttarget <= tfinish then goto loop
        end
rem A subroutine that delays "dwell" seconds:
delay:  t1 = time
        print "Pausing for ", dwell, " s."
delay1: if time - t1 < dwell * 1000 then goto delay1
        return
rem A subroutine that ramps from the current set point to "ttarget":
ramp:   itc601.tset = ttarget
        print "Ramping to ", ttarget, " K."
ramp1:  if itc601.isramping then goto ramp1
        return
```

Figure 71 TSEQ, an example macro which generates a sequence of temperature set point ramps and delays for ITC601.

6.5.1 Preparing to Use the IPS120-10 from ObjectBench

Figure 72 shows the IPS120-10 instrument driver main window which is displayed by double clicking on the IPS120-10 icon.

Before the software front panel can acquire data from the hardware instrument, you must connect (interface) the instrument to the PC and configure the instrument driver correctly as described in section 6.1.

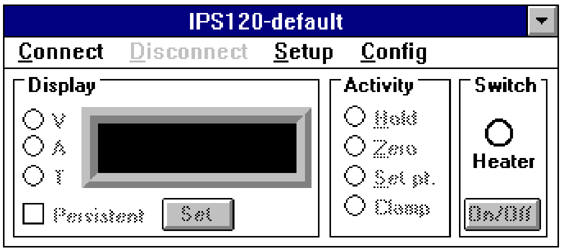


Figure 72 The IPS120-10 software front panel.

Finally, you may "connect" to the IPS120-10 by clicking on the IPS120-10's Connect menu option. The current field or current should appear on the display and be updated in real time. If this is not the case, an error message will be displayed which should give an indication of the nature of problem.

6.5.2 Operating the IPS120-10 from ObjectBench

The IPS120-10 software control panel is shown in Figure 72. It is very similar to the physical instrument front panel and operates in a similar way.

The "Display" box may be used to display field, current or power supply output voltage by clicking on the respective radio button. If the magnet has a superconducting switch fitted and the switch is closed, check the "Persistent" box to display the persistent field or current. The persistent values are based on the current present in the leads last time the superconducting switch was closed.

The "Activity" box contains radio buttons to switch the IPS120-10 into "Hold", "Zero", "Set Point" mode exactly as on the physical front panel of the IPS120-10. "Set Point" mode ramps the magnet to its set point, "Zero" mode ramps the magnet to zero current, and "Hold" mode freezes the IPS120-10 output at its present value. "Clamp" mode indicates that the PSU output is clamped (shorted) which is its power up state, or the result of a magnet quench.

If a superconducting switch is fitted to the magnet and the IPS120-10 has been correctly configured to drive it, the On/Off button in the "Switch" box is available for use. This button toggles the switch heater on and off, and the state of the heater is indicated by the red "LED" above the button. When the heater is on, the switch is resistive, while when the heater is off, the switch is superconducting and the magnet is persistent. You should allow several seconds, depending on the properties of the magnet itself, for changes to the switch heater to take effect.

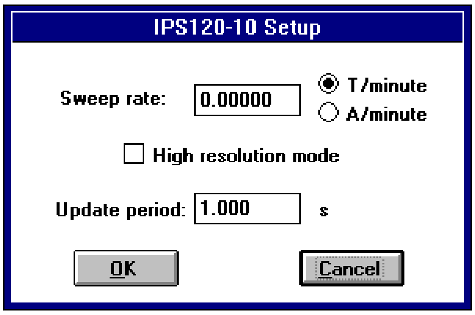
To enter the required field, click the "Set" button. This reveals a dialog box that allows you to enter the target field or current. If a negative current is set, the polarity changes. The power supply will sweep to the new set field, even if the polarity differs.

While one of the IPS120-10's front panel buttons is pressed by a user, the IPS120-10 delays responding to data requests by the PC. ObjectBench warns you when this situation occurs by sounding a two tone warning alarm through the PC's loudspeaker. This warning is a cue to the user to stop pressing the button! If the button continues to be pressed, ObjectBench will time out after 5 seconds.

When you have finished using the IPS120-10 from ObjectBench, click on the Disconnect main menu option to stop communication with the instrument.

6.5.3 The IPS120-10 Setup Dialog

Select the IPS120-10 "Setup" menu option to display the setup dialog as shown in Figure 73.



IPS120-10 Setup

Sweep rate: 0.00000

● T/minute

○ A/minute

□ High resolution mode

Update period: 1.000 s

Figure 73 The PS120-10 setup dialog.

The setup dialog allows you to specify the IPS120-10 sweep rate in tesla per minute or amps per minute. The maximum rate that may be used depends on the physical properties of your magnet. If too high a rate is selected, the power supply may voltage limit, or the magnet may quench. If in doubt, contact Oxford Instruments.

The setup dialog also allows the update period to be entered. This determines the rate at which new readings are acquired from the instrument and the display updated. A typical value is 2 seconds.

A feature of the IPS120 is its high resolution mode. When this feature is enabled by selecting the "High resolution mode" check box, an additional decimal place for field and current are displayed.

6.5.4 IPS120-10 Instrument Signals and Commands

Refer to section 7.10 for details of the signals and commands.

6.6 ILM Intelligent Level Meter

The Intelligent Level Meter model 200 (ILM200) is a family of helium and nitrogen level meters capable of monitoring up to three levels simultaneously. Its ObjectBench instrument driver allows cryogen levels and instrument status to be monitored. Full details of the instrument are included in a separate manual. The name of the ObjectBench driver is "ilm.drv".

6.6.1 Preparing to Use the ILM from ObjectBench

Figure 74 shows the ILM instrument driver main window which is displayed by double clicking on the ILM icon. Note that the appearance of the main window varies according to the channel configuration of the instrument.

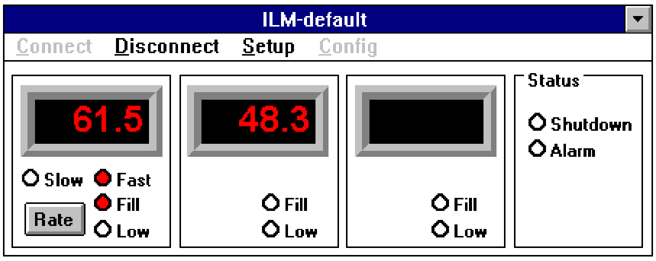


Figure 74 The ILM software front panel.

Before the software front panel can acquire data from the hardware instrument, you must connect (interface) the instrument to the PC and configure the instrument driver correctly as described in section 6.1.

Finally, you may "connect" to the ILM by clicking on the ILM's "Connect" menu option. The current level should appear on the display and be updated in real time. If this is not the case, an error message will be displayed which should give an indication of the nature of problem. The most likely cause of difficulty is in the interfacing options - if in doubt, review these first.

6.6.2 Operating the ILM from ObjectBench

The ILM software control panel is shown in Figure 74. It is very similar to the physical instrument front panel and operates in a similar way.

There are up to three level displays, which configure themselves automatically to correspond to the instrument configuration.

Beneath each readout are two lamps, labelled "Fill" and "Low", which indicate that auto fill (see ILM manual) is in progress and that the channel's level is low.

Beneath each Helium channel are two extra lamps and a button which can be used to control the channel's reading rate as on the actual instrument front panel. Note that under some circumstances, such as when in auto fill mode, only fast update is permitted.

In the status area, to the right of the dialog, are two lamps labelled "Rundown" and "Alarm". These correspond to the same lamps on the actual instrument, which indicate respectively that the magnet is being run down and that the instrument alarm is being sounded.

While one of the ILM's front panel buttons is pressed by a user, the ILM delays responding to data requests by the PC. ObjectBench warns you when this situation occurs by sounding a two tone warning alarm through the PC's loudspeaker. This warning is a cue to the user to remove his/her finger! If the button continues being pressed, ObjectBench will time out after 5 seconds.

When you have finished using the ILM from ObjectBench, click on the Disconnect main menu option to stop communication with the instrument.

6.6.3 The ILM Setup Dialog

The Setup dialog is accessed by selecting the instrument's "Setup" menu option. It allows the sampling rate to be set. The sampling period is the rate at which new readings are obtained from the instrument and the display updated. A typical value is 2 seconds.

6.6.4 ILM Instrument Signals and Commands

Refer to section 7.10 for details of the signals and commands.

6.7 Heliox Virtual Instrument

The Heliox virtual instrument is a virtual instrument driver that provides turnkey control of an Oxford Instruments Heliox ^{3}He insert. Automation of ^{3}He charge condensation allows effectively continuous operation of what is normally a single shot cryostat. As well as front panel buttons, macro language commands allows Heliox control to be incorporated very easily into your own programs.

6.7.1 Connecting the Heliox Virtual Instrument

A Heliox can be controlled using one, two or even more ITCs. There are two common configurations however:

- a single ITC with two sensors (for high and low temperatures) on the ^{3}He pot, a single sensor on the ^{3}He sorb and the heater output switchable between ^{3}He sorb and ^{3}He pot.
- two ITCs: The first ITC monitors ^{3}He pot sensors and has a heater output switchable between ^{3}He pot and ^{3}He sorb. The second ITC monitors and controls only the sorb temperature and heater.

Before it can be used, the Heliox virtual instrument must be connected to various signals from the ITC drivers. Additional output signals from the Heliox can then optionally be used in your own application. An example set of signal links is shown below for the two-ITC system described above.

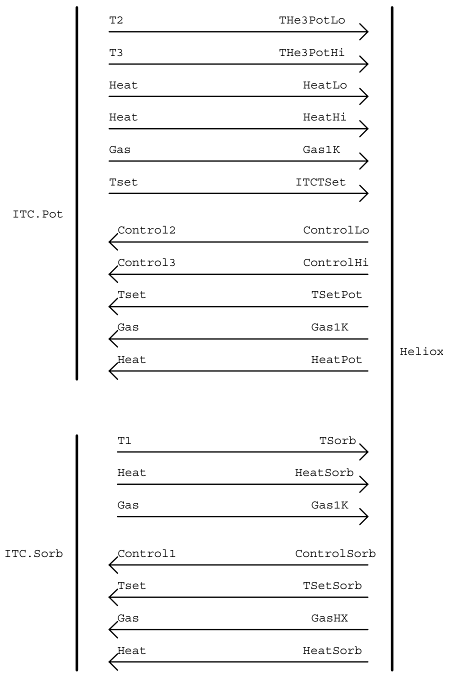


Figure 75 - Heliox signal links

In the case of a single ITC system, not all of these links will be used. Simply connect those that are available. The minimum requirements for the Heliox virtual instrument to be operational are:

- control over sorb heating (output signal heliox.xxx.ControlSorb must be connected)
- a sorb sensor (input signal heliox.xxx.TSorb must be connected)

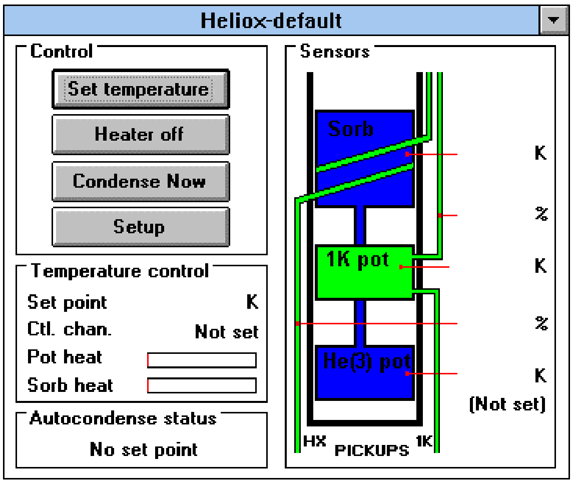
Automated valves are optional, but will improve the performance of the Heliox. The ability to adjust the 1K pot needle valve allows more complete condensation of the ^{3}He charge. Control over the heat exchanger gas flow will speed up the condensation process, and provide more flexibility in the range of operating temperatures. High temperature control is difficult with only a single ITC since the sorb temperature, and hence the thermal link between 1K pot and ^{3}He pot cannot be controlled.

- at least one sensor on the ^{3}He pot (both inputs heliox.xxx.THe3PotHi and heliox.xxx.THe3PotLo must be connected, but this can be to the same ITC output)

6.7.2 Using the Heliox Virtual Instrument

Having connected the Heliox virtual instrument as described above, you can use it to automate the operation of the cryostat. Figure 76 shows the main instrument dialog.

On the right of the dialog, a schematic diagram of a Heliox is used to display temperatures and gas flows. Starting from the top, these are the sorb temperature, the 1K needle valve setting, the 1K pot temperature, the heat exchange gas flow and the ^{3}He pot temperature. Below the ^{3}He pot temperature is the stability status. This can be either "Not set" (when the Heliox does not have a set point), "Stable" or "Stabilising". Stability is explained further in section 6.7.3.3. On the left of the dialog are the Control panel and two other panels showing further status information.



Heliox-default

Temperature control

Set point K

Ctl. chan. Not set

Pot heat

Sorb heat

Autocondense status

No set point

Figure 76 - Heliox virtual instrument

The buttons in the Control panel provide control over the Heliox. The Set Temperature button allows you to enter a new set point. The Setup parameters, described below, are used to decide how control is to be achieved at this temperature. Setting a low temperature may also trigger a recondensation of the ^{3}He as described in section 6.7.4. If the temperature entered is greater than the Maximum Heliox Temperature specified in the Setup, a dialog will alert you of the fact, and no further action will be taken.

The Heater Off button allows you immediately to turn off all heaters connected with the Heliox. This can be used if incorrect setup parameters have caused an excessive amount of heat to be applied to the Heliox. All heaters will be set to Manual control and zero output. To regain temperature control, you need to enter a Set Temperature again.

The Condense Now button unconditionally starts the ^{3}He recondensation procedure described in section 6.7.4.

6.7.2.1 The Setup button allows you to edit the Heliox setup information described in section 6.7.3.Control Panel

6.7.2.2 Temperature Control Panel

The Temperature Control panel shows how the Heliox temperature is being controlled. The Set Point is simply the temperature at which the Heliox virtual instrument has been asked to control. The Ctl. Chan. field shows the control channel being used to obtain this temperature. It can take on the values “Pot Low” or “Pot High” and “Sorb”. “Pot Low” uses sorb heating to control the pot temperature. “Pot High” uses a heater on the ^{3}He pot directly to control its temperature. ”Sorb” indicates that the sorb temperature is being directly controlled at a particular value. The Sample Heat and Sorb Heat fields simply show bar graphs of the relevant heater output, on a scale of 0 to 100 per cent.

The information on this panel relies on the Heliox virtual instrument knowing the true state of the ITCs used to control the Heliox itself. If you have used the Set Temperature button (or the Heliox.Tset signal) this will automatically be the case, provided your Signal Links have been correctly set up. When you first connect the ITCs used by the virtual instrument however, the Heliox obtains the set point from the ITC via the ITCTSet signal (see section 0). It is then assumed that the ITCs are configured as they would have been if this temperature had been entered using the Set Temperature button, and displays are updated accordingly. For example, if the set point from the ITC is 0.5K and this is below the Use He(3) Below temperature in the Setup, then the Heliox virtual instrument assumes that the sorb is being heated using feedback from the ^{3}He pot. The sorb heat display is therefore updated using values obtained from the HeatLo signal. If heat is being supplied to the sorb from a separate ITC this is not shown on the display.

If the Heliox has no set point then the sorb heater display is updated from both possible inputs (HeatLo and HeatSorb). If you have two ITCs and they have different heater outputs then the Sorb Heat display will fluctuate between these two values, since either could be supplying heat to the sorb. The Pot Heat will always show the value linked to HeatHi, which is usually the same as the HeatLo value.

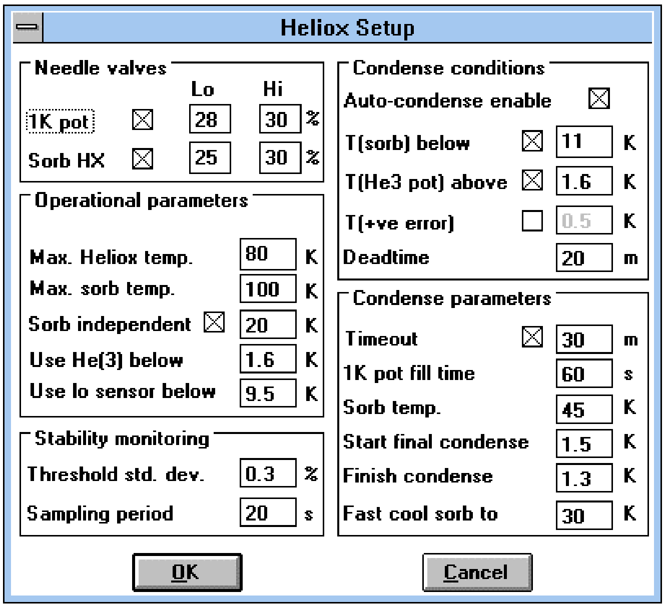
6.7.2.3 Autocondense Status

The Autocondense Status can display one of five messages depending on the configuration and status of the Heliox. Note that these conditions are checked for in the order shown, and subsequent conditions are not checked. If there is no set point for example, the message “No set point” is displayed and it is not possible to tell from the main dialog whether or not autocondensation has been enabled. The possible messages are given below:

- “No set point”
  The Heliox does not have a temperature set point and will therefore not perform an autocondense.
- “Set pt outside He(3) range”
  The set temperature is higher than the ^{3}He threshold value defined in the setup – autocondense will not be performed.
- “Condensing”
  The Heliox is condensing in the ^{3}He charge. The Condense dialog should also be visible.
- “Disabled”
  Autocondensation has been disabled from either the Setup dialog or the Autocondense signal.
- "Dead time: XXXs"
  Condensation has recently been performed and has been temporarily disabled for the duration specified by the Deadtime parameter in the Condense Conditions section of the Setup dialog. The time display counts down to zero at which point autocondensation is re-enabled.

6.7.3 Heliox setup dialog

The Heliox setup dialog allows you to specify the configuration of your Heliox and to control the way in which it operates. There are five sections to the setup – Needle Valves, Operational Parameters, Stability Monitoring, Condense Conditions and Condense Parameters.



Needle valves

		Lo	Hi	
1K pot	☒	28	30	%
Sorb HX	☒	25	30	%

Operational parameters

Max. Heliox temp.		80	K
Max. sorb temp.		100	K
Sorb independent	☒	20	K
Use He(3) below		1.6	K
Use lo sensor below		9.5	K

Stability monitoring

Threshold std. dev.	0.3	%
Sampling period	20	s

Condense conditions

Auto-condense enable	☒		
T(sorb) below	☒	11	K
T(He3 pot) above	☒	1.6	K
T(+ve error)	□	0.5	K
Deadtime		20	m

Condense parameters

Timeout	☒	30	m
1K pot fill time		60	s
Sorb temp.		45	K
Start final condense		1.5	K
Finish condense		1.3	K
Fast cool sorb to		30	K

Figure 77 - Heliox setup dialog

6.7.3.1 Needle Valves

The Needle Valves section allows you to specify whether or not you have automated needle valves for the two gas flow control points on a Heliox, and what values to use for them when setting a temperature.

The 1K pot needle valve is used to control the rate at which ^{4}He is sucked from the main bath into the 1K pot. This is an important parameter. If the rate is too low then the pot will eventually run dry and temperature control will rapidly be lost. If it is too high then performance will be decreased by the excessive inflow of relatively warm ^{4}He, and helium consumption will also be increased.

The Sorb HX valve controls the flow of helium gas past the ^{3}He sorb heat exchanger. This determines the cooling power on the sorb, and consequently the amount of heat required to maintain a given sorb temperature. There is a secondary effect in that too low a flow will create a large heat load on the 1K pot due to conduction of heat down the slow-moving column of gas.

Both valves have a check box that tells the software whether the valve is actually present. There are also two settings for each valve that are used when the Heliox is being operated at low and high temperatures respectively. Low temperatures are defined as those lower than specified in the Use He(3) Below parameter. These values will be greyed out if the associated check box is deselected. Having a 1K needle valve affects the way in which condensation proceeds (this is discussed in more detail in section 6.7.4). If there is no 1K needle valve, then the 1K Pot Fill Time and Finish Condense parameters will be greyed out since they are no longer relevant. Similarly, deselecting the Sorb HX valve will grey-out the Fast Cool Sorb To parameter.

6.7.3.2 Operational Parameters

These are general operating conditions for the Heliox.

The Max. Heliox Temp. and Max. Sorb Temp. parameters are provided as a safety measure. If either is exceeded at any time, the Heliox is put into manual control and all heat is switched off. The sorb temperature limit is particularly useful because of the possibility of thermal runaway, described in the Heliox manual. If the sorb heater is being used in an attempt to control the ^{3}He pot temperature at too high a value, then the sorb temperature can rise without limit while the pot temperature remains below its set point. This could ultimately cause damage to your system.

The Use He(3) Below value is used to specify a threshold temperature below which ^{3}He cooling is used, and above which ^{4}He cooling is used. In the former case, the ^{3}He sorb heater is used to control the ^{3}He pot temperature. In the latter case a heater on the pot is used directly to control its temperature.

Use Lo Sensor Below applies to systems with two sensors on the ^{3}He pot. Since there will generally be a region of overlap between the sensors, this specifies the temperature at which to make the changeover. Note that the low temperature sensor will generally saturate at some value and you should make sure that the threshold value you specify is below this since otherwise the high temperature sensor may never be selected.

6.7.3.3 Stability Monitoring

Stability monitoring allows a stability flag to be set if the standard deviation of the ^{3}He pot temperature falls below a specified limit.

The Threshold Std. Dev. parameter is used as the comparison value in deciding whether the temperature is stable. Not only the standard deviation, but also the difference between the ^{3}He pot temperature and the set point, must be equal to or lower than this value. Note that it is specified as a percentage of the pot temperature since, for example, 1mK may be a realistic limit at 0.5K but not at 50K.

The Sampling Period specifies the period in seconds over which the standard deviation is calculated. The number of points received in this time will depend on the update rate of the ITC in question. Because of performance considerations, the maximum number of points used is limited to 200. If more points than this are collected in the specified period, an error message will alert you to the fact. You should then reduce either the Sampling Period or the update rate of the ITC.

6.7.3.4 Condense Conditions

The Condense Conditions section allows you to define under what circumstances an auto-condense will be performed. There are three conditions, each of which can be enabled or disabled with a checkbox as well as being varied with a numerical value. Autocondensation can be completely disabled by deselecting the Auto-condense Enable checkbox.

During operation at ^{3}He temperatures, the Heliox's charge of ^{3}He is gradually boiled off from the pot and absorbed into the charcoal sorption pump. In order to maintain suction as it becomes saturated, the sorb temperature will lower steadily if temperature control of the sorb is being used to keep a constant pot temperature. At about 10K, maximum pumping is achieved and if the sorb temperature approaches this value it is a good indication that the ^{3}He charge is almost finished. This condition is checked with the T(sorb) Below parameter.

When the liquid ^{3}He is finished, the pot temperature will no longer be maintained, and will rise above the set point. This condition can be detected with the T(pot) Above parameter (which looks for the temperature exceeding a given absolute value) and T(+ve error) which looks for the pot temperature exceeding the set point by more than the specified value.

The Deadtime parameter allows a period of recovery after an autocondense has been performed. In re-establishing temperature control after condensing in the ^{3}He, the sorb temperature will often fall below 10K, and the ^{3}He pot temperature will oscillate about the set point. Since this could otherwise trigger an autocondense, the Deadtime specifies a period after condensation during which autocondensation is temporarily disabled.

6.7.3.5 Condense Parameters

The Condense Parameters section specifies the way in which condensation is performed. This is independent of how the condensation was triggered – whether by one of the conditions described above, or by a button press. Please see section 6.7.4 for a description of the actual recondensation process.

The Timeout parameter is used to detect a failure to recondense. If this option is selected, and condensation has not been successfully completed within the specified time, then the process is aborted. Note that recondensation will not timeout during the final stage of cooling the sorb since the actual condensation of ^{3}He has already finished at this stage.

On systems with an automated 1K needle valve, the condensation process is started by opening completely the needle valve in order to start with a full 1K pot. The 1K Pot Fill Time defines how long this is done for.

The Sorb Temp. specifies a sorb temperature to be maintained during condensation. The value typically used is about 45K. It should be high enough to ensure that nearly all the ^{3}He is driven out of the sorb. Note that during condensation some leeway is allowed on this temperature.

The Start Final Condense value defines the pot temperature at which the 1K needle valve (if automated) is closed down prior to the final condense. If the needle valve is not automated, then this is the end of condensation, otherwise Finish Condense defines a further pot temperature at which condensation is deemed to have finished.

Finally, on systems with an automated heat exchanger needle valve, the valve is opened completely until the sorb temperature has dropped below the Fast Cool Sorb To parameter. This is purely a time-saving measure which allows the sorb to reach its normal control temperature more quickly than would otherwise be the case.

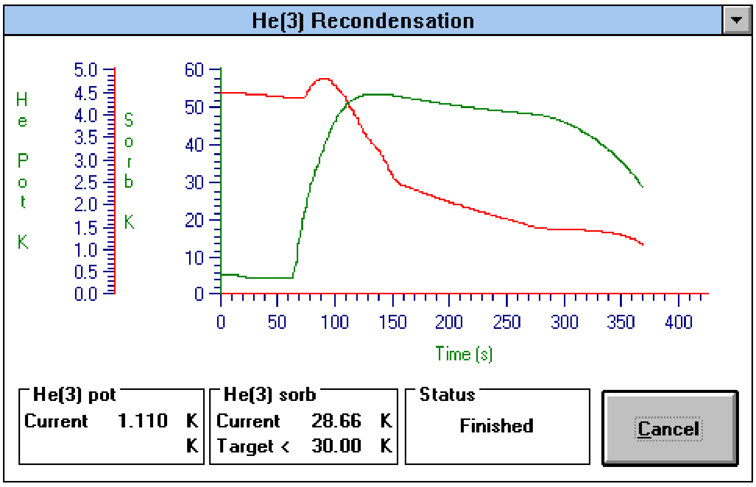
6.7.4 Recondensation

Recondensation is the process of liquefying the ^{3}He charge of the Heliox prior to using it for cooling the cryostat below 1K or so. This section describes the process as implemented in the software.

Efficient recondensation involves achieving the lowest possible ^{3}He pot temperature combined with the highest possible ^{3}He sorb temperature. When this has been achieved, the maximum possible fraction of the Heliox's ^{3}He charge will exist as liquid in the ^{3}He pot. This will give the longest possible hold time subsequently. In practice, it is found that there is little to be gained from sorb temperatures in excess of 40K or so. The objective therefore is simply to obtain the lowest possible pot temperature while controlling the sorb at about 45K. Since different systems will perform differently, the obtainable pot temperatures will vary. In determining which values to use for your system, you must strike a balance between achieving both long hold times and quick, reliable recondensation.

Figure 78 shows the dialog that appears during recondensation. The graph plots pot and sorb temperatures on different scales, and these temperatures are also displayed numerically in panels below. The current activity is shown in the Status panel, and a Cancel button allows the process to be aborted.

Before recondensation begins, you are given a few seconds to cancel out of the dialog before any action is taken. This feature is provided to save you having to wait for the temperature to restabilise if you have accidentally triggered condensation, either by pressing the Condense button or through incorrect setup parameters.



He(3) pot
Current 1.110 K
K

He(3) sorb
Current 28.66 K
Target < 30.00 K

Status
Finished

Figure 78 - ^{3}He recondensation

If there is an automated 1K pot needle valve, the first step is to open it fully for the time specified by the 1K Pot Fill Time Parameter to allow the pot to fill with helium liquid. The needle valve is then set to its low temperature value. The sorb is set to control at the temperature specified in the setup – usually about 45K – while waiting for the pot temperature to fall below the Start Final Condense value. Once this has been achieved, and the sorb is close to its set point, the first stage of condensation has been completed. If there is an automated 1K pot needle valve, this is then closed completely. The new target then becomes the Finish Condense value. In the absence of an automated 1K pot needle valve however, sorb cooling is performed immediately after the Start Final Condense value is reached.

Sorb cooling is the final stage of the condense procedure. It consists simply of opening the heat exchange valve completely until the sorb temperature falls below the value specified by the Fast Cool Sorb To parameter. This stage saves a significant amount of time compared with waiting for the sorb to cool with normal gas flow, and allows your experiment to continue as soon as possible. It can only be performed with an automated heat exchange valve. If there is no such valve, this step is skipped.

The steps involved in the recondensation process are summarised in Figure 79.

After condensation has finished, temperature control is resumed at the current set point, and the valves are returned to values appropriate to that temperature as defined in the setup. If you Cancel out of the condensation dialog, the heaters are turned off, and the needle valves set to their low temperature values. The main dialog will show “No set point” and further recondensations will be disabled until you reset the temperature. If the condensation times out, a message will appear in the condensation dialog, and the dialog will remain until you press Cancel.

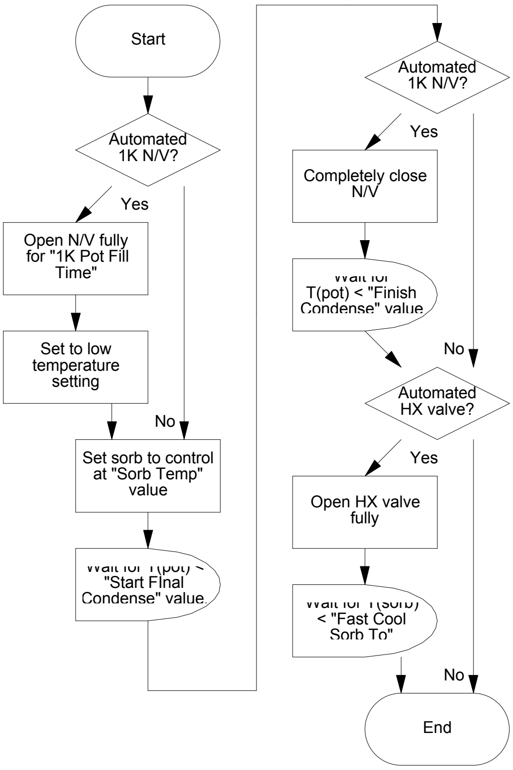
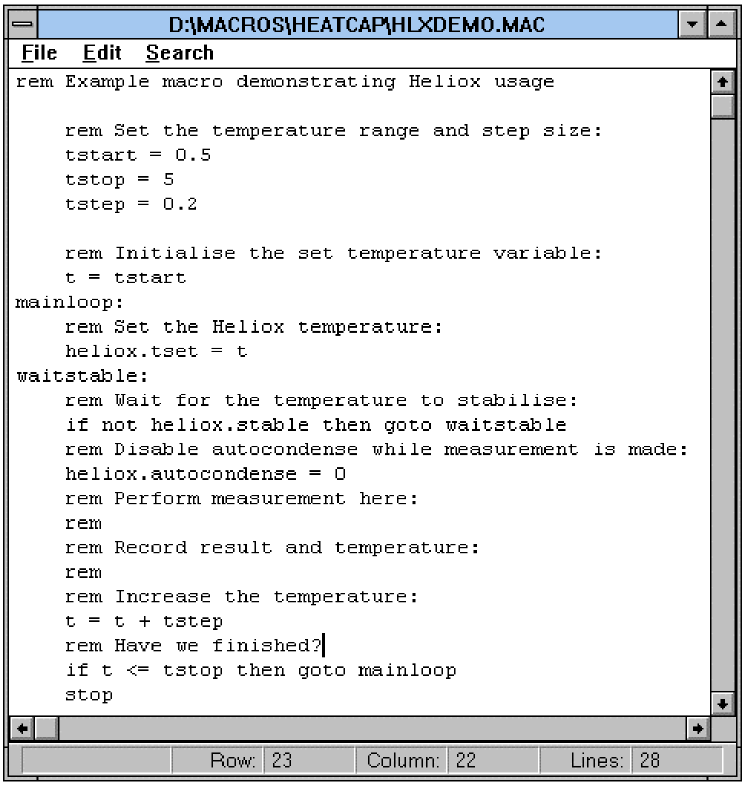

Start
Automated 1K N/V?
Yes
Open N/V fully for "1K Pot Fill Time"
Set to low temperature setting
No
Set sorb to control at "Sorb Temp" value
Wait for T(pot) < "Start Final Condense" value
Automated 1K N/V?
Yes
Completely close N/V
Wait for T(pot) < "Finish Condense" value
No
Automated HX valve?
Yes
Open HX valve fully
Wait for T(sorb) < "Fast Cool Sorb To"
No
End

Figure 79 - Recondensation process

6.7.5 Usage Notes

This section gives some guidelines for using the Heliox driver from your own macro language programs.

A typical cryogenic experiment consists of setting a temperature, waiting for the temperature to stabilise, performing a measurement and then recording the temperature and result. Figure 80 gives an outline of how such an experiment might be implemented using the macro language and the Heliox driver.



```
rem Example macro demonstrating Heliox usage

    rem Set the temperature range and step size:
    tstart = 0.5
    tstop = 5
    tstep = 0.2

    rem Initialise the set temperature variable:
    t = tstart
mainloop:
    rem Set the Heliox temperature:
    heliox.tset = t
waitstable:
    rem Wait for the temperature to stabilise:
    if not heliox.stable then goto waitstable
    rem Disable autocondense while measurement is made:
    heliox.autocondense = 0
    rem Perform measurement here:
    rem
    rem Record result and temperature:
    rem
    rem Increase the temperature:
    t = t + tstep
    rem Have we finished?
    if t <= tstop then goto mainloop
    stop
```

Figure 80 - Example macro

The key points illustrated by the macro are:

- The Heliox temperature is set with the simple command **heliox.tset = t**, independently of how this temperature is achieved (whether by ^{3}He or ^{4}He cooling). Achieving this temperature may involve condensing the ^{3}He charge.
- A simple two line loop delays measurements until the set temperature is reached and the temperature has stabilised. Achieving the set temperature may involve a change in control mode, or even a recondensation, but the macro language does not have to know about these things – the only effect is to increase the time taken.
- Recondensation is disabled during measurements to prevent an autocondense and then re-enabled afterwards. Note that if recondensation is immediately triggered upon enabling it, the next set point is stored by the condensation dialog and set on completion.

6.7.6 Heliox Instrument Signals and Commands

Refer to section 7.10.7 for details of the signals and commands.

7 Macro Language Reference

This section provides a complete reference to the macro language, complementing the informal introduction in section 4.5.

7.1 Constants

Numerical constants are accepted in the normal format which allows an optional sign, decimal point and exponent. Examples of valid numeric constants are 2, **3.1415927, -3, 6E4** and **-3.0E-2**. Hexadecimal numbers may be used if they are terminated with a lower case "**h**", for example, "**300h**" is 768 in decimal.

String constants consist of text between quotation marks. For example, "**test**", "**x**" and "**this is a string**" are examples of string constants. The maximum length of a string constant is 160 characters.

7.2 Variables

Numerical variables are automatically created as required, and may consist of any length of letters and digits. The first character must be a letter. All variables represent double precision floating point numbers which may take values in the range -1.7E-308 to 1.7E308, with 15 digits of internal precision. Upper case and lower case in variable names is not distinguished so for example **lim, LIM** and **Lim** all refer to the same variable.

String variables are named in exactly the same way as numerical variables, with the addition of a suffix **"$".** For example, **a$** and **name$** are string variables. There is no limit on the length of a string variable (but see the section on string constants). You can include line feed and carriage return characters in strings respectively as “\n” and “\r”.

Instrument variables are actually pseudo-variables that implicitly write and read data to and from specific instrument drivers. The general format for an instrument variable is as follows X.Y.Z, where X is the instrument name, Y is its configuration name and Z is the signal name. Y is used to distinguish between different instances of the same instrument, and may be omitted if there is only one instance. When an instrument variable appears on the left side of an equals sign, it performs a write of the value to the instrument; when it appears on the right side, it performs a read from the instrument.

Specific signal names for instrument signals depend on the instrument type and are listed in other sections of this manual. In general instrument signals have a combination of three characteristics: **read**, **write**, and **update**. **Read**able signals may be read at any time by appearing on the right of an equals sign. **Write**able signals may be written to at any time by appearing on the left of an equals sign. **Update** signals may invoke an action whenever a new value is available, such as calling an **on signal** handler in a macro program or triggering an XY plot window point. Refer to the section of this manual on the instrument in question for details of signals and their characteristics.

The following are examples of valid instrument variable names for an ITC503 instrument:

itc503.t1	(if there is only one ITC503 in the system)
itc503.vsm.t1	(if there is an ITC503 with configuration "vsm")

There are certain keywords which may not be used as variable or label names. They are listed below.

abs acos and append asin atan close connect
constant cos dim disconnect end exp file
get GPIBread$ gpibread$ GPIBreceive$ gpibreceive$
GPIBtransmit gpibtransmit GPIBwrite gpibwrite
graph hex$ if include input int ioread ioreadword
iowrite iowriteword left$ len ln load
log log10 mid$ new not off okbox on
on open or overwrite pi ppoll print
read return right$ run screen serialclose serialopen
serialread$ serialwrite setGPIBineos setgpibineos
setGPIBouteos setgpibouteos setgpibtimeout setGPIBtimeout
setserialineos setserialouteos setserialtimeout share signal
sin spoll sqrt stop str$ tan then time
timer true val write xor yesnobox

Figure 81 ObjectBench macro language reserved keywords.

7.3 Arrays and Data Files

The ObjectBench macro language supports one and two dimensional arrays. Arrays must be dimensioned before use as in BASIC. An array may be resized by redimensioning it, but its previous data is lost. For example, to create an array of 100 elements and assign a value to its tenth element:

```
dim x(100)
x(10) = 45.67
```

ObjectBench format data files can be read into structures that behave like arrays by using the **read** command as follows:

```
read “data.dat”, y
print y.desc$
print y.time$
print y.rows, y.columns
print y.units$(1), y.parameter$(1)
print y(2, 1)
```

The file data is read into a structure named **y**. The description and time from the file are printed, followed by the number of rows and columns of data in the file. The units and parameter name for column 1 are printed, and finally the value at row 2 column 1 is printed.

7.4 Expressions

Numerical variables constants may be combined in algebraic expressions in which arithmetic operations and brackets have the conventional meanings and precedence. An expression may appear in any position that a simple constant or variable value can appear. The following are examples of valid expressions using the arithmetic operators +, -, *, \ and **:

```
1.2
x
x * 1.2
(a + 4) * (b - 3) / (c + 1)
-x ** 3
(a - 3.3 * (b + 1))
itc503.t1 / itc503.t2 - 4
```

There are also bitwise Boolean operators available for bit manipulation. They are: "&" (bitwise AND), "|" (bitwise OR), "^" (bitwise exclusive OR), and "~" (bitwise NOT or complement). Numbers may be bit shifted using the left shift operator ("<<") and the right shift operator (">>"). The following examples should clarify this:

1 << 2	evaluates to 4
3 >> 1	evaluates to 1
x = x ^ 4	inverts bit 2 of x
x = (x & FFh) | 1	filters out bits 0-7 of x and sets bit 1

Instrument variables may appear in expressions as long as they have the readable characteristic described in the previous section. In the last example above, itc503 channel 1 and channel 2 temperatures are read and divided. Note that this only reads the most recent value of t1 and t2: if the instrument sampling period is set to a high value, the reading may actually be out of date. It is possible to force a fresh hardware read of a value by using the keyword read, as follows:

```
read itc503.t1
read itc503.t1 - read itc503.t2
```

String variables and constants may also be combined, but only using the operator '+'. For example, valid string expressions are:

```
"test"
a$
a$ + "test"
```

Boolean expressions occur in if..then statements. A Boolean expression is a combination of a number of numerical expressions, comparison operators (=, < , >, <=, >=, <>) and logical operators (and, or, xor, not) in the conventional way. Examples of valid Boolean expressions are as follows:

```
x > 1
(x + 2) / y <> 5
x < 5 and g = 2
```

Caution should be exercised when comparing values using the "=" operator, because rounding in the internal floating point representation can create small errors. For example,

(2 / 3) * 3 = 2

would evaluate to FALSE.

7.5 Assignment

Variables may be assigned values from expressions in the usual way using the "=" operator. The general form is:

variable = *expression*

where *variable* and *expression* must have the same type (numerical or string) and expression is defined in the previous section. *Variable* may be an instrument variable only if it has the writeable characteristic.

7.6 Labels

A label may precede any statement or occur on a line on its own. It is simply an identifier like a variable name followed by a ":". The label may serve as a destination for a goto, a gosub, or the <ESCAPE> key handler. Examples of labels are:

loop:
l1:
escape:

The label **escape:** has a special meaning. If it is present, when the user attempts to interrupt the macro by pressing <ESCAPE>, execution will jump to the label as if a goto had been encountered. Pressing <ESCAPE> a second time always terminates execution. This feature can be used to implement "tidying up" operations, such as stopping a field ramp or resetting parameters to quiescent operating values.

7.7 Instrument Commands

Instruments may support specific commands which are detailed in the section in this manual for that instrument. To execute an instrument command, append the command text to the end of the instrument identifier as follows:

itc503.vsm.auto
itc503.manual

The first example is a statement which executes the command **auto** using instrument **itc503** with configuration name **vsm**. If there is only one ITC503 in the system, the configuration name may be omitted, as in the second example.

7.8 Macro Command and Function Reference

This section contains an alphabetic list of macro commands that may be used in immediate mode (in the macro shell window) and in macro program files. Macro commands and functions must always appear in lower case. Multiple commands may appear on one line of a macro file if they are separated using ";".

Note that additional commands are listed in section 7.9.1.

abs(*expression*) This function returns the value of expression without its sign, that is it always returns a positive number.

acos(*expression*) Returns the inverse cosine of the value in radians.

asin(*expression*) Returns the inverse sine of the value in radians.

atan(*expression*) Returns the inverse tangent of the value in radians.

close #n Closes any opened output channels for channel #n.

cos(*expression*) Returns the cosine of the value, which should be supplied in radians.

dim *ident*(*expression*) Dimension an array for use. One and two dimensional arrays are supported. Arrays may be redimensioned, but all previous data in the array is discarded. All array elements are initialised to zero.

exp(*expression*) Returns the exponent of the supplied value.

format #*n, value, value*..

This statement defines a data format for output to a file, a graph or the screen. *value* is a numeric expression, which may optionally be followed by ["*label*", "*units*"], where *label* is the name of the signal and *units* are its units. If the latter are included, they will be included in the data file and used to label axes and plots. *n* is a unique channel identifier which is used again in **open** and **close** statements.

gosub *label* The **gosub** statement transfers execution to the line with the label. *label* does not include the final ":". When a **return** statement is encountered in the subroutine, execution resumes at the line following the original **gosub** statement.

goto *label* The **goto** statement transfers execution to the line with the label. label does not include the final ":".

hex$(*expression*) This function converts the supplied value to a string, using hexadecimal representation.

if *boolexp* **then <goto|gosub>** *label*

If the Boolean expression is TRUE, the **goto** or **gosub** label is executed. No statement other than **goto** or **gosub** may follow the **if..then** clause.

include "*fname*"

The contents of the file *fname* are conceptually copied into the program at this point. This feature is useful when common subroutines are kept in separate file. There is a nesting limit of ten files.

input *strexpression*, x

Displays a dialog box which uses the string supplied to prompt the user to enter a value for the variable x. The string *expression* may be omitted.

left$(*strexpression*, n)

Returns a string which is the first n bytes of the string supplied.

len(*strexpression*)

This function returns the length of a string.

ln(*expression*)

Returns the natural logarithm of *expression*.

load "*progname*"

Loads a macro program from a file ready for running. Load actually tokenises the input stream and prepares to run the program. Syntax errors are reported at this stage. *progname* is the name of the file containing the macro program. If no path is specified, the default path set with the main ObjectBench menu option Setup/Directories is used.

log(*expression*)

Returns the natural logarithm of the value.

log10(*expression*)

Returns the base 10 logarithm of the value.

mid$(*strexpression*, n1, n2)

Returns the part of the string supplied starting at position n1 and of length n2. n2 may be omitted, in which case the remainder of the string starting at n1 is returned.

new

Clears any loaded program and all variables.

on signal on

Activates **on signal** handlers. By default, signal handlers are not activated.

on signal off

Deactivates **on signal** handlers. By default, signal handlers are not activated.

on signal *signal*

This statement precedes a subroutine which will be invoked whenever a new value for signal *signal* is available. The subroutine must finish with a **return** statement in the usual way. **On signal** handlers must be activated using **on signal** on before the subroutine can be invoked.

open #n, file = "*fname*", "*desc*"**[, overwrite]**

This statement opens a data file for output in the format defined in the **format** statement with the same number. *fname* is the name of the data file to open, *desc* is some descriptive text which is included in the data file. If the file already exists, a simple dialog box is displayed asking whether you wish to overwrite the existing file. Supplying the keyword **overwrite** forces any existing file to be overwritten. You may supply a DOS wildcard character in the filename if you wish to select a file from a scrolling list at run time. For example **"open #1, file = "*.dat""** displays a list of files matching "*.dat" at run time, and the user may select one or enter a new file name.

You can specify comments to be saved with the data in the file by assigning a value to the variable **filecomment$** before the file is opened.

open #*n*, **screen** This statement causes data to be written to the macro shell window in accordance with the **format** #*n* statement.

open #*n*, **graph =** "*title*", **[*overwrite*]**

This statement creates a graph window ready to receive data in the format described by the **format** #*n* statement. *title* is the title to be used for the window. If the *overwrite* keyword is present, any existing window with the same title will be used.

You can specify comments to be displayed above the data in the graph by assigning a value to the variable **graphcomment$** before the file is opened.

print *arg, arg...* Prints the argument list in the macro shell. Multiple arguments are separated with commas, which do not add spaces to the output. There may be any number of arguments, including zero, and each argument may be either a numeric or string expression. ObjectBench always starts a new line after a print statement.

read *strexpression, ident* Opens and reads the contents of a named ObjectBench data file into the named structure. The file name may contain wildcard characters (* and ?) in which case a standard file open dialog is displayed. Once a data file has been read using this command, data from the file can be accessed as follows:

ident.desc$ The file description.

ident.time$ The file time.

ident.rows The number of rows of data.

ident.columns
The number of columns of data.

ident.parameter$(n)
The parameter name for the nth column.

ident.units$(n)
The units for the nth column.

ident(r, c)
The data value at row r, column c.

rem
Marks this entire line as a comment, which is ignored by the computer.

return
The **return** statement should be the last statement in a subroutine. It resumes execution from the line following the original **gosub** statement. If it occurs in any other context, a run-time error occurs.

right$(*strexpression*, n)
Returns a string which is the last n bytes of the string supplied.

run
Executes the macro program most recently loaded, if any.

sin(*expression*)
Returns the sine of the value, which should be supplied in radians.

stop
Program execution unconditionally ceases and any open data channels are automatically closed.

str$(*expression*)
Converts a numerical variable to a string. For example, a$ = str$(3.4) assigns the string "3.4" to a$. *expression* is a numerical expression as described in section 7.3.

sqrt(*expression*)
Returns the square root of the value.

tan(*expression*)
Returns the tangent of the value, which should be supplied in radians.

time
This variable always contains the number of milliseconds that have elapsed since Windows was started.

val(*strexpression*)
Returns the value of a string. The string must be a number such as "1.234" or "5e-3".

write #*n*
The **write** statement writes data in the format defined by the **format** #*n* statement to the data channels opened, that is any combination of a disk file, a graph or the screen.

yesnobox(*strexpression*)
Display a dialog box containing the text supplied, and buttons for "Yes", "No" and "Cancel". It returns 1 if the user clicks on "Yes", 0 if the user clicks on "No". Clicking on "Cancel" is equivalent to pressing <ESCAPE> while a macro is running, that is it attempts to jump a label "escape:". If there is no such label, the macro terminates.

7.9 Controlling Instruments From Other Suppliers

ObjectBench includes drivers for all Oxford Instruments instruments, and a few instruments from other suppliers which are required for specific systems such as VSMs and Faraday Balances. If you need to acquire data from any other instrument, you can use generic RS232 serial or GPIB commands which form part of the macro language. In addition, you can interface using direct I/O access to expansion boards installed in the PC. Access to these interfaces is via additional macro language commands, listed below.

If you wish to use RS232 interfacing, you must make sure that there is a free RS232 port on your PC.

Note that ObjectBench supports GPIB interface boards which are compatible with National Instruments' PCII, PCIIa and AT TNT. Please refer to the readme.txt file for more details.

7.9.1 Additional Macro Commands for Interfacing

The following macro language commands are available in addition to those described in section 7.8, and may be used for communication with instruments not supplied by Oxford Instruments.

gpibwrite(*address, string*)
Write the string supplied to the gpib address supplied.

gpibread$(*address*)
Read a string from the gpib instrument whose address is supplied.

ioread(*address*)
Read and return the value from the I/O port at the supplied address on the PC's expansion bus.

iowrite(*address, value*)
Write the value supplied to the I/O port at the supplied address on the PC's expansion bus.

serialclose(*comport*)
Close a serial port after having been opened by **serialopen**. This argument supplied is the com port number.

serialopen(*openstring*)
Open a serial port for low level I/O using **serialwrite** and **serialread**. The string supplied is in the same format as for BASIC. For example, **“serialopen("com1:9600,n,8,2")”** opens com1for 9600 baud, no parity bit, 8 data bits and 2 stop bits.This statement should have a matching **serialclose** statement.

serialread$(*comport*)
Read input from the serial port whose number is supplied. The results is returned as a string.

serialwrite(*comport,text*)
Write the text supplied to the comport supplied.

setgpibouteos(*term*)
Set the terminating character to be appended to data written to a gpib instrument by supplying its ASCII value.

setgpibineos(*term*) Set the terminating character to expect appended to data read from a gpib instrument by supplying its ASCII value.

setgpibtimeout(*timeout*) Set the gpib input timeout for reading data, in ms.

spoll(*address*) Serial poll the gpib instrument whose address is supplied and return the result.

setserialtimeout(*timeout*) Set the serial port input timeout for reading data, in ms.

setserialouteos(*term1,term2*) Set the terminator or terminators to append to text written to a serial port. The values supplied are the ASCII codes for the characters to use.

setserialineos(*term*) Set the terminator to expect at the end of serial input data. The value supplied is the ASCII code for the terminating character.

ppoll() Parallel poll all gpib instruments and return the result.

7.10 Instrument Signal and Command Names

This section contains listings of the signals and commands available for each instrument described in section 6.

7.10.1 ITC502 Instrument Signals and Commands

The following commands and signals are provided by the ITC502 instrument driver for use in macro programs.

Commands:

itc502.connect Equivalent to clicking on the ITC502 "Connect" menu option.

itc502.disconnect Equivalent to clicking on the ITC502 "Disconnect" menu option.

itc502.auto Puts the ITC502 in auto temperature control mode.

itc502.manual Puts the ITC502 in manual temperature control mode.

itc502.ramp/itc502.step Puts the ITC502 into ramp or step mode. Future set point changes will result in a ramp or a step respectively. The ramp rate is set using the ramprate variable described below.

itc502.X Issues the command X on the ISOBUS addressed to the ITC502, and checks that the reply does not contain "?". This command is included for testing purposes only.

Signals:

itc502.tset	The set point temperature.
itc502.t1, t2, t3	Channel 1, 2 and 3 measured temperatures.
itc502.heat	The heater output in %.
itc502.gas	The gas flow output in %.
itc502.p, i, d	The PID control terms.
itc502.control	The control channel, 1-3.
itc502.period	The display updating period in seconds.
itc502.ramprate	The rate to use for set point ramping (K/min).
itc502.isramping	Value 1 if the set point is ramping, 0 otherwise.

Signal characteristics:

	Write	**Read**	**Update**
itc502.tset	X	X	
itc502.t1, t2, t3		X	X
itc502.heat	X	X	X
itc502.gas	X	X	X
itc502.p, i, d	X	X	
itc502.control	X	X	
itc502.period	X	X	
itc502.ramprate	X		
itc502.isramping		X	

7.10.2 ITC503 Instrument Signals and Commands

The following commands and signals are provided by the ITC503 instrument driver for use in macro programs.

Commands:

itc503.connect	Equivalent to clicking on the ITC503 "Connect" menu option.
itc503.disconnect	Equivalent to clicking on the ITC503 "Disconnect" menu option.
itc503.auto	Puts the ITC503 in auto temperature control mode.
itc503.manual	Puts the ITC503 in manual temperature control mode.
itc503.step/itc503.ramp	Puts the ITC503 into ramp or step mode. Future set point changes will result in a ramp or a step respectively. The ramp rate is set using the ramprate variable described below.

itc503.manualpid, itc503.tablepid, itc503.autopid

Set the PID mode to auto, manual, or table respectively.

See section 6.3.2 for more information on PID modes.

itc503.X	Issues the command X on the ISOBUS addressed to the ITC503, and checks that the reply does not contain "?". This command is included for testing purposes only.

Signals:

itc503.tset	The set point temperature.
itc503.t1, t2, t3	Channel 1, 2 and 3 measured temperatures.
itc503.heat	The heater output in %.
itc503.gas	The gas flow output in %.
itc503.p, i, d	The PID control terms.
itc503.control	The control channel, 1-3.
itc503.period	The display updating period in seconds.
itc503.ramprate	The rate to use for set point ramping (K/min).
itc503.isramping	Value 1 if the set point is ramping, 0 otherwise.
itc503.overshoot	Overshoot requested for auto tuning.
itc503.threshold	Temperature noise threshold for auto tuning.
itc503.control1	A flag indicating that channel 1 is the control channel.
itc503.control2	A flag indicating that channel 2 is the control channel.
itc503.control3	A flag indicating that channel 3 is the control channel.

Signal characteristics:

	Write	Read	Update
itc503.tset	X	X	
itc503.t1, t2, t3	X	X	
itc503.heat	X	X	X
itc503.gas	X	X	X
itc503.p, i, d	X	X	
itc503.control	X	X	
itc503.period	X	X	
itc503.ramprate		X	
itc503.isramping	X		
itc503.overshoot	X		
itc503.threshold	X		
itc503.control1	X	X	
itc503.control2	X	X	
itc503.control3	X	X	

7.10.3 ITC601 Instrument Signals and Commands

The following commands and signals are provided by the ITC601 instrument driver for use in macro programs.

Commands:

itc601.connect	Equivalent to clicking on the ITC601 "Connect" menu option.
itc601.disconnect	Equivalent to clicking on the ITC601 "Disconnect" menu option.
itc601.auto	Puts the ITC601 in auto temperature control mode.
itc601.manual	Puts the ITC601 in manual temperature control mode.

itc601.step/itc601.ramp Puts the ITC601 into ramp or step mode. Future set point changes will result in a ramp or a step respectively. The ramp rate is set using the ramprate variable described below.

itc601.manualpid, itc601.tablepid

Set the PID mode to manual or table respectively.

itc601.X Issues the command X on the ISOBUS addressed to the ITC601, and checks that the reply does not contain "?". This command is included for testing purposes only.

Signals:

itc601.tset The set point temperature.

itc601.t The measured temperature.

itc601.heat The heater output in %.

itc601.p, i, d The PID control terms.

itc601.control The control channel, always 1.

itc601.period The display updating period in seconds.

itc601.ramprate The rate to use for set point ramping (K/min).

itc601.isramping Value 1 if the set point is ramping, 0 otherwise.

Signal characteristics:

	Write	Read	Update
Itc601.tset	X	X	
Itc601.t		X	X
Itc601.heat	X	X	X
Itc601.p, i, d	X	X	
Itc601.control		X	
Itc601.period	X	X	
Itc601.ramprate	X		
Itc601.isramping		X	

7.10.4 IPS120-10 Instrument Signals and Commands

The following commands and signals are provided by the IPS120 instrument driver for use in macro programs.

Commands:

ips120.connect	Equivalent to clicking on the IPS120 "Connect" menu option.
ips120.disconnect	Equivalent to clicking on the IPS120 "Disconnect" menu option.
ips120.hold, ips120.zero, ips120.setpt	Equivalent to clicking on the IPS120 Hold, Zero, or Set point radio buttons respectively.
ips120.forward ips120.reverse	Equivalent to clicking on the IPS120 Forward or Reverse radio buttons.
ips120.heateroff	Turn the superconducting switch heater on or
ips120.heateron	off, if fitted.
ips120.X	Issues the command X on the ISOBUS addressed to the IPS120, and checks that the reply does not contain "?". This command is included for testing purposes only.

Signals:

ips120.h	The existing field (read) or target field (write).

ips120.i The existing current (read) or target current (write).

ips120.ramping Is the field ramping ?

ips120.rh The field ramp rate in tesla per minute.

ips120.ri The current ramp rate in amps per minute.

ips120.period The display updating period in seconds.

Signal characteristics:

	Write	Read	Update
ips120.h	X	X	X
ips120.i	X	X	X
ips120.ramping	X		
ips120.rh	X	X	
ips120.ri	X	X	
ips120.period	X	X	

7.10.5 ILM Instrument Signals and Commands

The following commands and signals are provided by the ILM instrument driver for use in macro programs.

Commands:

ilm.connect Equivalent to clicking on the ILM "Connect" menu option.

ilm.disconnect Equivalent to clicking on the ILM "Disconnect" menu option.

ilm.fast1, ilm.fast2, ilm.fast3 Set channel 1-3 to update in fast mode. Only applicable to helium level channels.

ilm.slow1, ilm.slow2, ilm.slow3 Set channel 1-3 to update in slow mode. Only applicable to helium level channels.

ilm.X Issues the command X on the ISOBUS addressed to the ILM, and checks that the reply does not contain "?". This command is included for testing purposes only.

Signals:

ilm.L1, ilm.L2, ilm.L3 Read the level of channel 1-3 as a percentage.

ilm.shutdown Is the shutdown lamp lit?

ilm.alarm Is the alarm lamp lit?

ilm.period The display updating period in seconds.

Signal characteristics:

	Write	Read	Update
ilm.L1, L2, L3		X	
ilm.shutdown		X	
ilm.alarm	X		
ilm.period	X	X	

7.10.6 SMC4 (VSM) Instrument Signals and Commands

The following commands and signals are provided by the SMC4 instrument driver for use in macro programs.

Commands:

smc4.connect	Equivalent to clicking on the SMC4 "Connect" menu option.
smc4.disconnect	Equivalent to clicking on the SMC4 "Disconnect" menu option.
smc4.go	Drive the SMC4 axes to the destination set up by assigning the target positions to the smc4.x, smc4.y, smc4.z and smc4.r signals.
smc4.stop	Unconditionally stop any axis movements in progress.
smc4.lock	If Z compensation is enabled for the instrument, lock Z at the current reading by dynamically compensating the Z motor position using the Z transducer.
smc4.unlock	Deactivate Z compensation.
smc4.X	Issues the command X on the ISOBUS addressed to the SMC4, and checks that the reply does not contain "?". This command is included for testing purposes only.

Signals:

smc4.x, smc4.y, smc4.z, smc4.r	The current axis positions (if read) or the axis target positions (if written).
smc4.rx, smc4.ry, smc4.rz, smc4.rr	The rates at which to drive the axes to their destinations in physical units per second. Note that the rate is rounded down to the nearest permissible value appearing in the Setup dialog.
smc4.busy	Is any axis running at the moment ? Has value 0 or 1.

smc4.isx, smc4.isy smc4.isz, smc4.isr	Returns 1 if the axis is present in the configuration, 0 otherwise
smc4.period	The display updating period in seconds.

Signal characteristics:

	Write	Read	Update
smc4.period	X	X	
smc4.x,y,z,r	X	X	X
smc4.rx,ry,rz,rr	X	X	
smc4.busy		X	
smc4.isx,isy,isz,isr		X	

7.10.7 Heliox Instrument Signals and Commands

The following commands and signals are provided by the Heliox instrument driver. Some are necessary for Heliox operation, some are for use in macro programs.

Commands

heliox.condense	Initiates the condensation process

Signals:

THe3PotLo	The low temperature ^{3}He pot sensor value
THe3PotLo	The high temperature ^{3}He pot sensor value
THe3Pot	The temperature of the ^{3}He pot
T1KPot	The temperature of the 1K pot
TSorb	The temperature of the ^{3}He sorb
TSet	The Heliox (^{3}He pot) set point
ITCTSet	The Heliox (^{3}He pot) set point read from the ITC
TSetSorb	The sorb temperature set point
CtlHe3PotLo	Control of the sorb heater from the ^{3}He pot sensor
CtlHe3PotHi	Control of the pot heater from the ^{3}He pot sensor
CtlSorb	Control of the sorb heater from the sorb sensor
Flow1K	The 1K pot needle valve setting
FlowHX	The ^{3}He sorb heat exchange needle valve setting
Busy	The Heliox status (recondensing or not)

Stable	The stability status of the Heliox temperature
AutoCondense	The auto-condense status of the Heliox (enabled or disabled)
THe3PotHi	The high-temperature ^{3}He pot sensor value
THe3PotLo	The low-temperature ^{3}He pot sensor value
HtrOP	The Heliox control heater output
HeatLo	The low-temperature control heater output
HeatHi	The high-temperature control heater output
HeatSorb	The sorb heater output

Signal Characteristics

	Write	**Read**	**Update**
THe3PotHi	X		
THe3PotLo	X		
THe3Pot		X	X
T1KPot	X	X	X
TSorb	X	X	X
Tset	X	X	
TSetPot		X	
ITCTSet	X		
TSetSorb		X	
ControlLo	X		
ControlHi	X		
ControlSorb	X		
HeatLo	X		
HeatHi	X		
HeatPot		X	X
HeatSorb	X	X	X
Gas1K	X	X	X
GasHx	X	X	X
Busy		X	X
Stable		X	X
AutoCondense	X	X	X
HtrOP		X	X
HeatLo	X		
HeatHi	X		
HeatSorb	X	X	X

8 How to use Application Dialogs

ObjectBench macro programs offer a way of customising ObjectBench to your specific needs by creating a set of macros that perform your most common tasks. Once you have these macros, you can created a convenient user interface for invoking them. This is achieved by using a third party Resource Editor to create a standard Windows dialog box, with buttons that call the named macros.

8.1 Loading Application Dialogs

Application dialogs are contained in files with the extension ".app". To load an application dialog into ObjectBench, use the main menu "Windows"/"Applications" option. This will present a list of the application files present and available for loading.

8.2 Creating Application Dialogs

This section assumes greater knowledge of the use of and workings of Windows than the rest of this manual. It should be read in conjunction with the Borland Resource Workshop manual.

To create your own custom application dialog, you need to obtain a copy of the Borland Resource Workshop and install it on your PC according to its own documentation. Resource Workshop (RW) is a very powerful and flexible tool for editing Windows resources (dialogs, icons, bitmaps, etc.), and is supplied with its own excellent documentation. Creation of custom application dialogs only uses a small part of its capabilities, so this section should not be considered as complete reference to RW. Instead, only those features used when creating application dialogs are included.

For simplicity, creation of custom dialogs is presented as a series of steps, as follows.

Use the default custom application file supplied ("default.app") in the \OB directory as a starting point. Use DOS or Windows to copy it to a file with a different name, which will become your custom application dialog.

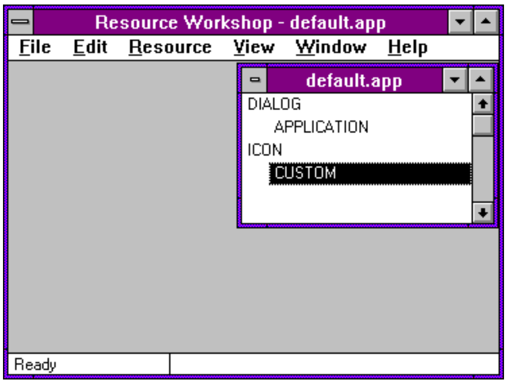

Resource Workshop - default.app
File Edit Resource View Window Help
default.app
DIALOG
APPLICATION
ICON
CUSTOM
Ready

Figure 82 The Resource Workshop, showing the contents of the default custom application.

Run RW from the Windows Program Manager. Click on the "File"/"Open project" menu option. On the "Open Project" dialog box, use the "File type" drop-down list to select "DLL library". Then type the name of your application dialog file into the "File name" box, including its path and ".app" extension. RW will display a list of the resources in your application file in a window on the right of the display, shown in Figure 82.

Double click on "Application" in the list. The application will be displayed in a window ready for editing together with some editing tools. The default application dialog is shown in Figure 83. It is now ready for customising as described in the following sections. **Take care throughout not to change any of the settings of the default custom dialog or delete any of the default controls shown, or it will not behave correctly.**

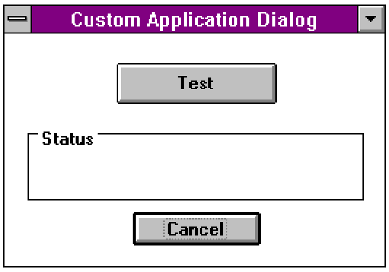


Figure 83 The default custom application dialog ready for editing.

Customise the dialog to suit your application. First change the title to suit your needs by double clicking on it. You may then add the following features to the dialog using RW:

Add a button by selecting the "Controls"/"Button" menu item and clicking the mouse at the position you require the button at. Resize and move the button as required with the mouse. Double click on button and change its text as required - on the same dialog, set the button's ID to a **unique** value **greater than 100**. When this application is loaded into ObjectBench, clicking on the button will invoke the macro with the same name as the button text (stripped of spaces etc.). If you require a different macro to be called, use RW to create a "string resource" whose ID is the same as the button's, and whose text is the name of the macro required.

Add a parameter entry field by selecting the "Controls"/"Edit text" menu item and clicking the mouse at the position required. Move and resize the field with the mouse. Double click on the field and set its ID to a **unique** value **greater than 100**. Use RW to create a "string resource" with the same ID, and set its text to the name of a macro variable (numerical or string). When the application dialog is loaded into ObjectBench and executed, the value the user enters in this field will be assigned to the macro variable named in the string resource.

Add text to the dialog by selecting the "Controls"/"Static text" menu option and clicking the mouse at the required position. Move or resize it as before, and double click on it to enter the text you require. Leave its ID set to -1.

Radio buttons and check boxes are added in the same way as parameter fields. Use to mouse to select the appropriate control from the "Controls" menu and click at the required position. Double click on the control and set its ID to a unique value greater than 100. On the same dialog, be sure to select **auto** check box or **auto** radio button. Add corresponding entries to the string resource table as before. The macro variables named here will be set to 0 or 1 depending on whether the user has selected the check box or radio button.

If you wish to invoke a subsidiary dialog box from a button, use RW to create a new dialog resource with required **title**. Use the "Resource"/"Rename" menu option to set its **name** to the text on the calling button or the string resource entry for the button (if any). Double click on the dialogs menu bar and set its "class" to "obapp", making sure "system menu" and "modal frame" are checked, and choose a font of size 8. Add controls to this dialog in exactly the same way as for the calling dialog. Dialogs may be nested to any depth.

If you wish to design a special icon for your application, double click on the "Custom" entry in the list shown in Figure 82. You may now use RW to edit the default icon.

When the application dialog is used in ObjectBench, and a macro is run by clicking on a dialog button, it may be cancelled by clicking on the "Cancel" button with the mouse. This is equivalent to pressing escape in the macro shell, and invokes an escape handler (if any) as described in section 7.6.

When the custom application dialog is complete, save it and test it by loading it into ObjectBench as described in section 8.1. It is generally easiest to develop your macros in the macro shell before attempting to invoke them from a custom application dialog.

9 Appendix I: Data File Format

All ObjectBench data files created by macros and XY plotting windows may be read in by an analysis window. These are ASCII text files in the format described below. This information is useful if you wish to import data from a different ASCII file format by modifying it with a text editor

```
Tue Oct 06 14:59:59 1992
Experiment 1
          time,      x,                 y
          ms,        -,                 -
           7:0       7:0                7:0
2.4478040E+07   0.0000000E+00   0.0000000E+00
2.4478095E+07   1.0000000E+00   0.0000000E+00
2.4478095E+07   2.0000000E+00   0.0000000E+00
2.4478095E+07   3.0000000E+00   0.0000000E+00
2.4478095E+07   4.0000000E+00   0.0000000E+00
2.4478095E+07   5.0000000E+00   0.0000000E+00
2.4478150E+07   6.0000000E+00   0.0000000E+00
2.4478150E+07   7.0000000E+00   0.0000000E+00
2.4478150E+07   8.0000000E+00   0.0000000E+00
2.4478150E+07    9.0000000E+00  0.0000000E+00
2.4478150E+07   1.0000000E+01   0.0000000E+00
2.4488311E+07   0.0000000E+00   0.0000000E+00
```

Figure 84 An ObjectBench data file.

The first six lines are reserved for special information. The remainder of the file contains a number of columns of numbers as shown, and may be of any length required. There may be any number of columns, although if the data is to be plotted on a graph there must obviously be at least two columns.

The first line is the date which is automatically added by ObjectBench when the file is created.
The second line is descriptive text entered by the user, of any length.
The third line consists of a list of signal names, separated by commas.
The fourth line consists of a list of units names, separated by commas.
The fifth line is not used but reserved for future use. It should be left blank.
The sixth line consists of a list of format codes matching P:Q as shown. P is the number of significant figures in the data. Q is not yet used and should be set to zero.

Comment lines may be included at any point in the first six lines by preceding them with a ";" character. Note that the number of entries in third and fourth lines must be the same as the number of columns of data. If this is not true, ObjectBench will refuse to load the data file.